```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
```

FILEID**DUTUSUBS

```
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU    SSSSSSSS  UU      UU  BBBBBBBB      SSSSSSSS
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU    SSSSSSSS  UU      UU  BBBBBBBB      SSSSSSSS
DD      DD  UU      UU      TT      UU      UU  SS          UU      UU  BB      BB  SS
DD      DD  UU      UU      TT      UU      UU  SS          UU      UU  BB      BB  SS
DD      DD  UU      UU      TT      UU      UU  SS          UU      UU  BB      BB  SS
DD      DD  UU      UU      TT      UU      UU  SS          UU      UU  BB      BB  SS
DD      DD  UU      UU      TT      UU      UU    SSSSSS    UU      UU  BBBBBBBB      SSSSSS
DD      DD  UU      UU      TT      UU      UU    SSSSSS    UU      UU  BBBBBBBB      SSSSSS
DD      DD  UU      UU      TT      UU      UU          SS  UU      UU  BB      BB          SS
DD      DD  UU      UU      TT      UU      UU          SS  UU      UU  BB      BB          SS
DD      DD  UU      UU      TT      UU      UU          SS  UU      UU  BB      BB          SS
DD      DD  UU      UU      TT      UU      UU          SS  UU      UU  BB      BB          SS
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  SSSSSSSS    UUUUUUUUUU  BBBBBBBB    SSSSSSSS
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  SSSSSSSS    UUUUUUUUUU  BBBBBBBB    SSSSSSSS
```

```
LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

(2) 327 DECLARATIONS
(2) 416 IRP - CDRP Consistancy Check
(3) 459 ----- INITIALIZATION/REINITIALIZATION ROUTINES -----
(3) 460 DUTU$CREATE_CDDB - Create and initialize a CDDB
(4) 718 DUTU$POLL_FOR_UNITS - Poll a server for its available units
(5) 1020 DUTU$UNITINIT - Class driver unit initialization routine
(6) 1094 DUTU$FAILOVER_UCB - Failover a single UCB
(6) 1259 DUTU$FAILOVER_IODB - Update DDB I/O database after a failover
(7) 1363 ----- NEW UNIT ROUTINES -----
(7) 1364 DUTU$NEW_UNIT - Process a possible new unit
(8) 1696 DUTU$LOOKUP_UCB - Locate a given MSCP unit on given CDDB chain
(9) 1747 DUTU$SETUP_DUAL_PATH - Reflect dual path access in I/O database
(10) 1920 DUTU$FIND_DDB - Find or create a DDB with the right DEVNAM
(11) 2023 DUTU$LINK_SEC_UCB - Link UCB to secondary DDB chain
(12) 2078 DUTU$LINK_UCB2CDDB - Link a UCB into a CDDB chain
(13) 2134 DUTU$SEVER_SEC_UCB - Unlink a secondary UCB
(14) 2175 DUTU$SEVER_CDDB - Sever UCB from CDDB chain
(15) 2229 Default device name and device type tables
(16) 2296 DUTU$GET_DEVNAM - Form new unit device name, DDCn
(17) 2430 DUTU$GET_DEVTYPE - Translate media-id to a device type
(18) 2502 DUTU$INIT_CONN_UCB - Initialize connection dependent UCB fields
(19) 2540 DUTU$SETUP_CDP_UCB - Setup remote/local dual pathed device
(20) 2684 ----- CANCEL ROUTINES -----
(20) 2685 Cancel-CDRP Definitions
(21) 2738 DUTU$CANCEL - Cancel I/O Routine for class drivers
(22) 2960 SEND_ABORTS - Send ABORT commands for active MSCP requests
(23) 3023 DUTU$CHECK_NOCANCEL - Check for no similar cancel requests
(24) 3086 DUTU$BUILD_CANIO_CDRP - Allocate & Initialize a cancel-CDRP
(25) 3169 DUTU$CANCEL_RDTWAIT - Cancel a thread on the RDT wait queue
(26) 3214 DUTU$CANCEL_RDT - Cancel a thread holding a RSPID
(27) 3309 DUTU$TEST_CANCEL_DONE - Test for completed cancel operation
(28) 3355 DUTU$DISCONNECT_CANCEL - Do disconnect cleanup for cancels
(29) 3412 DUTU$END_CANCEL - Finish a cancel operation
(29) 3413 DUTU$CLEANUP_CANCEL - Cleanup a cancel operation
(30) 3477 DUTU$TEST_CANCEL_CDRP - Should a CDRP be canceled
(31) 3519 ----- GENERAL SUPPORT ROUTINES -----
(31) 3520 DUTU$RESET_MSCP_MSG - Reset MSCP command packet
(32) 3563 DUTU$INIT_MSCP_MSG_UNIT - Initialize a MSCP message w/ unit number
(32) 3564 DUTU$INIT_MSCP_MSG - Initialize a MSCP message
(33) 3639 DUTU$SEND_DRIVER_MSG - Send driver internal message to MSCP server
(33) 3640 DUTU$SEND_MSCP_MSG - Send command message to MSCP server
(34) 3688 DUTU$INTR_ACTION_XFER - Interpret transfer action table
(34) 3689 DUTU$INTR_ACTION_N - Interpret non-transfer action table
(35) 3806 DUTU$RESTORE_CREDIT - Restore allocated message credit
(36) 3874 DUTU$INSERT_RESTARTQ - Insert outstanding CDRP in restart queue
(37) 3982 DUTU$RECONN_LOOKUP - Reconnection SCS Lookup Action Routine
(38) 4070 DUTU$DRAIN_CDDB_CDRPQ - Drain Active CDRPs Queue
(39) 4113 DUTU$TERMINATE_PENDING - Fail pending requests with SS$_VOLINV
(40) 4176 DUTU$DEALLOC_ALL - Deallocate all SCS resources
(40) 4177 DUTU$DEALLOC_RSPID_MSG - Deallocate SCS RSPID and msg. buf.
(41) 4222 DUTU$POST_CDRP - Queue CDRP for post processing
(42) 4302 DUTU$KILL_THIS_THREAD - Fix thread caught by async. conn. failure
(43) 4356 DUTU$CHECK_RWAITCNT - Validate UCB$W_RWAITCNT
(44) 4417 DUTU$LOG_IVCMD - Error log an invalid MSCP command
(45) 4698 DUTU$DODAP - Do Determine Access Paths Processing
(46) 4761 DUTU$SEND_DUPLICATE_UNIT - Send duplicate unit message to operator
(47) 4792 "Register" Dump Routines
(47) 4793 o DUTU$DUMP_COMMAND - Copy MSCP command to diagnostic buffer
(47) 4819 o DUTU$DUMP_ENDMESSAGE - Copy MSCP end message to diagnostic buffer
(48) 4850 DUTU$FILL_MSCP_MSG - Zero fill a partial MSCP message

M 1

```
0000     1          .TITLE  DUTUSUBS DISK/TAPE CLASS DRIVER SUBROUTINES
0000     2          .IDENT  'V04-001'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ;
0000    31 ; FACILITY:
0000    32 ;
0000    33 ;       MSCP Disk and Tape Class Drivers
0000    34 ;
0000    35 ; ABSTRACT:
0000    36 ;
0000    37 ;       This module contains subroutines used by both the MSCP disk and tape
0000    38 ;       class drivers.
0000    39 ;
0000    40 ; ENVIRONMENT:
0000    41 ;
0000    42 ;       This module is linked into both the DUDRIVER and the TUDRIVER.  The
0000    43 ;       routines herein are called by those drivers as needed.  See individual
0000    44 ;       routine descriptions for interface details.
0000    45 ;
0000    46 ;--
0000    47 ;
0000    48 ; AUTHOR: Ralph O. Weber,       CREATION DATE: 7-AUG-1983
0000    49 ;
0000    50 ; MODIFIED BY:
0000    51 ;
0000    52 ;       V04-001 ROW0416         Ralph O. Weber          14-SEP-1984
0000    53 ;               Make DUTU$NEW_UNIT ignore requests to create a unit 4095.
0000    54 ;               This will prevent creating units for sprious messages from the
0000    55 ;               HSC.
0000    56 ;
0000    57 ;       V03-039 ROW0412         Ralph O. Weber          21-AUG-1984
```

```
0000     58 ;                Change bugcheck issued when more than three paths are found
0000     59 ;                for a single device for INCONSTATE to IVDSKCONFG, invalid disk
0000     60 ;                configuration.  This is a temporary measure to make the error
0000     61 ;                condition more clear and is expected to be fixed better in a
0000     62 ;                future version.
0000     63 ;
0000     64 ;        V03-038 ROW0404         Ralph O. Weber          23-JUL-1984
0000     65 ;                Fix race condition between 2P device configuration and unit
0000     66 ;                failover.  This eliminates attempts to failover units which
0000     67 ;                are not fully dual-path configured.
0000     68 ;
0000     69 ;        V03-037 ROW0401         Ralph O. Weber          21-JUL-1984
0000     70 ;                Change DUTU$CREATE_CDDB so that attempts to have two CDDBs for
0000     71 ;                the same system-id on the same device class (disk or tape)
0000     72 ;                result in the second CDDB and all related structures being
0000     73 ;                deallocated back to pool and no second connect attempt taking
0000     74 ;                place.  This eliminates exteranous paths to a device in the
0000     75 ;                event that SYSGEN or a user of SYSGEN attempts to "connect"
0000     76 ;                multiple times to the same remote server.
0000     77 ;
0000     78 ;                Clear the DEV$V_AVL bit in all CDP UCBs.  This prevents
0000     79 ;                allocating of or assigning a channel to such devices.  Thus,
0000     80 ;                the devices become even more completely unusable which is the
0000     81 ;                desired goal.
0000     82 ;
0000     83 ;        V03-036 ROW0400         Ralph O. Weber          21-JUL-1984
0000     84 ;                Enhance DUTU$FAILOVER_UCB so that changes to the I/O database
0000     85 ;                which must be done with write access to the I/O database mutex
0000     86 ;                are done in a separate fork thread.  This fork thread uses a
0000     87 ;                separate fork block which is part of the CDDB which the
0000     88 ;                unit(s) are failing away from.  This eliminates the need to
0000     89 ;                have write access to the I/O database mutex before calling
0000     90 ;                DUTU$FAILOVER_UCB and prevents stalling I/O requests due to
0000     91 ;                lack of access to the mutex.  Since such requests can no
0000     92 ;                longer stall, it is no longer possible to get a deadlock when,
0000     93 ;                for example, a process holding the mutex must perform a page
0000     94 ;                fault read in order to complete and release the mutex.
0000     95 ;
0000     96 ;        V03-035 ROW0395         Ralph O. Weber          20-JUL-1984
0000     97 ;                Setup use of DAP CDRP by DUTU$POLL_FOR_UNITS.  Also build
0000     98 ;                correct coordination of DAP CDRP usage between
0000     99 ;                DUTU$POLL_FOR_UNITS and DUTU$DODAP.
0000    100 ;
0000    101 ;        V03-034 ROW0390         Ralph O. Weber          20-JUL-1984
0000    102 ;                Fix DUTU$GET_DEVNAM so that the controller letter is the one
0000    103 ;                choosen by SYSGEN.  Previously, "A" was used, regardless of
0000    104 ;                what was correct.
0000    105 ;
0000    106 ;        V03-033 ROW0388         Ralph O. Weber           8-JUL-1984
0000    107 ;                Add adjustment of wait counter for status of newly used
0000    108 ;                connection to DUTU$FAILOVER_UCB.  This function formerly was
0000    109 ;                performed by DU_END_MNTVER, but some race conditions exist
0000    110 ;                which require that the adjustment be made after every
0000    111 ;                successful failover.
0000    112 ;
0000    113 ;        V03-032 ROW0387         Ralph O. Weber           7-JUL-1984
0000    114 ;                Add DUTU$RECONN_LOOKUP, the broken connection processing
```

```
0000   115 ;                action routine for both the RSPID-wait lookup SCS service and
0000   116 ;                the RDT lookup SCS serivce.  Add DUTU$DRAIN_CDDB_CDRPQ, the
0000   117 ;                routine which removes all entries in the CDDBs active CDRP
0000   118 ;                queue and places them in the restart queue.
0000   119 ;
0000   120 ;        V03-031 ROW0380         Ralph O. Weber          22-JUN-1984
0000   121 ;                Add error recovery for case where DUTU$NEW_UNIT call in
0000   122 ;                DUTU$POLL_FOR_UNITS cannot allocate memory for new UCB.  Error
0000   123 ;                recovery is to restart pool from the beginning.
0000   124 ;
0000   125 ;        V03-030 ROW0364         Ralph O. Weber          17-MAY-1984
0000   126 ;                Hack DUTU$POST_CDRP to adjust wait count when IO$_PACKACK is
0000   127 ;                posted with UCB$V_MSCP_PKACK set, or when IO$_SETCHAR or
0000   128 ;                IO$_SETMODE are posted with UCB$V_TU_SEQNOP set.  This is an
0000   129 ;                outragous hack, but it cannot be done correctly until after
0000   130 ;                FT 2 ships.
0000   131 ;
0000   132 ;        V03-029 ROW0362         Ralph O. Weber           7-MAY-1984
0000   133 ;                Fix stack usage bug in DUTU$LOG_IVCMD.  We were pushing more
0000   134 ;                onto the stack than we were poping off.
0000   135 ;
0000   136 ;        V03-028 ROW0361         Ralph O. Weber           5-MAY-1984
0000   137 ;                Add DUTU$DODAP, the common disk/tape class driver routine for
0000   138 ;                sending Determine Access Path commands.  This version of the
0000   139 ;                DAP thread is designed not to have multiple thread forking on
0000   140 ;                one CDRP.
0000   141 ;
0000   142 ;        V03-027 ROW0360         Ralph O. Weber           5-MAY-1984
0000   143 ;                Add testing of UCB$V_MSCP_PKACK to DUTU$CHECK_RWAITCNT.
0000   144 ;
0000   145 ;        V03-026 LMP0235         L. Mark Pilant,         17-Apr-1984  13:59
0000   146 ;                Add a template ORB that is hooked up to the template UCB
0000   147 ;                just prior to creating a new UCB and ORB from the templates.
0000   148 ;
0000   149 ;        V03-026 ROW0344         Ralph O. Weber          11-APR-1984
0000   150 ;                > Change DISKCLASS and some INCONSTATE bugchecks to MSCPCLASS.
0000   151 ;                > Correct method for obtaining I/O database access in
0000   152 ;                  DUTU$FIND_DDB to prevent race condition believed responsible
0000   153 ;                  for multiple DDBs per controller letter.
0000   154 ;                > Correct fork thread handling bugs in DUTU$RESTORE_CREDIT.
0000   155 ;
0000   156 ;        V03-025 ROW0339         Ralph O. Weber           8-APR-1984
0000   157 ;                Add generalized "invalid command" processing support.
0000   158 ;                Specifically, add DUTU$LOG_IVCMD.
0000   159 ;
0000   160 ;        V03-024 ROW0338         Ralph O. Weber           7-APR-1984
0000   161 ;                Add support for DO_ACTION macro in the form of two subroutines,
0000   162 ;                DUTU$INTR_ACTION_XFER and DUTU$INTR_ACTION_N.
0000   163 ;
0000   164 ;        V03-023 ROW0337         Ralph O. Weber           7-APR-1984
0000   165 ;                Alter LINK_NEW_UCB to account for disappearance of
0000   166 ;                UCB$L_CANLINK.
0000   167 ;
0000   168 ;        V03-022 ROW0335         Ralph O. Weber           4-APR-1984
0000   169 ;                Add DUTU$UNITINIT, a unit initialization routine which deletes
0000   170 ;                the bogus UCBs created by SYSGEN and still keeps useful UCBs
0000   171 ;                after a power failure.
```

```
0000   172 ;
0000   173 ;       V03-021 ROW0334         Ralph O. Weber          3-APR-1984
0000   174 ;               Add initialization of UCB$L_MAXBCNT based upon value found in
0000   175 ;               PDT$L_MAXBCNT to LINK_NEW_UCB and LINK_2P_UCB.
0000   176 ;
0000   177 ;       V03-020 ROW0333         Ralph O. Weber          30-MAR-1984
0000   178 ;               Correct DUTU$RESET_MSCP_MSG to handle the possibility that
0000   179 ;               RECYCL_MSG_BUF may discover a broken connection.  This is done
0000   180 ;               by transfering to DUTU$KILL_THIS_THREAD whenever and error is
0000   181 ;               encountered.
0000   182 ;
0000   183 ;       V03-019 ROW0331         Ralph O. Weber          26-MAR-1984
0000   184 ;               Move all major CANCEL support routines to DUTUSUBS.  Also add
0000   185 ;               some sanity checks in DUTU$INSERT_RSTARTQ and DUTU$POST_CDRP.
0000   186 ;
0000   187 ;       V03-018 ROW0329         Ralph O. Weber          22-MAR-1984
0000   188 ;               Eliminate toggling of UCB$V_FLOVR in DUTU$FAILOVER_UCB.  The
0000   189 ;               information is no longer used else where in the class drivers.
0000   190 ;               Also all failover if destination path is CDDB$V_RECONNECT or
0000   191 ;               CDDB$V_RSTRTWAIT.  Because of some connection failure
0000   192 ;               scenarios, it is important to allow a failover back to a
0000   193 ;               previously tested path when that path is still reconnecting
0000   194 ;               but has a working connection.
0000   195 ;
0000   196 ;       V03-017 ROW0327         Ralph O. Weber          21-MAR-1984
0000   197 ;               Change DUTU$SETUP_CDP_UCB so that it never declares a class
0000   198 ;               driver UCB to be a CDP UCB.
0000   199 ;
0000   200 ;       V03-016 ROW0325         Ralph O. Weber          19-MAR-1984
0000   201 ;               Remove references to UCB$L_MVIOQFL and UCB$L_MVIOQBL.  This
0000   202 ;               queue header is no longer used.  Also change system device
0000   203 ;               unit number check in DUTU$POLL_FOR_UNITS to use UCB$W_MSCPUNIT
0000   204 ;               instead of UCB$W_UNIT.
0000   205 ;
0000   206 ;       V03-015 ROW0312         Ralph O. Weber          26-FEB-1984
0000   207 ;               Change DUTU$GET_DEVTYPE to return a device type equivalent to
0000   208 ;               the device name choosen by DUTU$GET_DEVNAM when UCB$L_MEDIA_ID
0000   209 ;               is zero.
0000   210 ;
0000   211 ;       V03-014 ROW0309         Ralph O. Weber          23-FEB-1984
0000   212 ;               Get CDDB address setup for call to DUTU$SETUP_CDP_UCB in
0000   213 ;               LINK_NEW_UCB.  On that same page, stop using R2 as the UCB
0000   214 ;               base address.  Use R5 instead.  R2 gets corrupted by
0000   215 ;               DUTU$SETUP_CDP_UCB.
0000   216 ;
0000   217 ;       V03-013 ROW0304         Ralph O. Weber          12-FEB-1984
0000   218 ;               Fix three bug discovered while testing the new mount verifica-
0000   219 ;               tion oriented failover:
0000   220 ;               - Do not allow failover to a CDDB with CDDB$V_RSTRTWAIT set.
0000   221 ;               - Copy new CRB address into a failed-over UCB.
0000   222 ;               - If IOC$LINK_UCB reports a duplicate unit number in
0000   223 ;                 LINK_NEW_UNIT, deallocate the working UCB.  Clearly, two
0000   224 ;                 threads got started for the same unit and the current one
0000   225 ;                 got there last.  This change also requires that all
0000   226 ;                 operations which "link" the working UCB to the rest of the
0000   227 ;                 I/O database MUST occur after the call to IOC$LINK_UCB.
0000   228 ;
```

```
0000   229 ;       V03-012 ROW0300         Ralph O. Weber          9-FEB-1984
0000   230 ;               Add toggling of UCB$V_MSCP_FLOVR to TUTU$FAILOVER_UCB.
0000   231 ;
0000   232 ;       V03-011 ROW0298         Ralph O. Weber          9-FEB-1984
0000   233 ;               Setup use of CDRP$W_ENDMSGSIZ to hold the size of an incomming
0000   234 ;               sequenced message.  This replaces use of CDRP$L_IOST2+2 whose
0000   235 ;               use causes valuable input information to be overwritten.
0000   236 ;
0000   237 ;       V03-010 ROW0297         Ralph O. Weber          7-FEB-1984
0000   238 ;               Add test for UCB$V_TU_SEQNOP as reason to bump wait count in
0000   239 ;               DUTU$CHECK_RWAITCNT.
0000   240 ;
0000   241 ;       V03-009 ROW0295         Ralph O. Weber          6-FEB-1984
0000   242 ;               Add DUTU$CHECK_RWAITCNT, a routine which validates
0000   243 ;               UCB$W_RWAITCNT and bug checks if it is wrong.
0000   244 ;
0000   245 ;       V03-008 ROW0293         Ralph O. Weber          4-FEB-1984
0000   246 ;               Setup use of INIT_MSCP_MSG in DUTU$POLL_FOR_UNITS.  Remove
0000   247 ;               references to the unused CDDB$L_CONNQFL and CDDB$L_CONNQBL.
0000   248 ;
0000   249 ;       V03-007 ROW0286         Ralph O. Weber          22-JAN-1984
0000   250 ;               Add the following routines:
0000   251 ;               > DUTU$TEST_CANCEL_CDRP which tests a CDRP and determines
0000   252 ;                 whether or not it should be canceled.
0000   253 ;               > DUTU$KILL_THIS_THREAD which temporarily terminates a
0000   254 ;                 thread caught by an inopportune connection failure.
0000   255 ;               > DUTU$INSERT_RESTARTQ which handles first pass inserting
0000   256 ;                 CDRPs into the restart queue during connection failure
0000   257 ;                 processing.
0000   258 ;
0000   259 ;       V03-006 ROW0284         Ralph O. Weber          19-JAN-1984
0000   260 ;               Change handling of CDDB$L_ORIGUCB to guarantee that it is the
0000   261 ;               system device UCB and to hand craft a mount verification style
0000   262 ;               IO$_PACKACK IRP which brings the unit online.  It might be
0000   263 ;               argued that this code belongs only in the disk class driver.
0000   264 ;               However, with the plans to permit booting from a MSCP tape
0000   265 ;               disucssed recently over lunch, it seems unwise to prohibit
0000   266 ;               tapes from performing this function.
0000   267 ;
0000   268 ;       V03-005 ROW0279         Ralph O. Weber          14-JAN-1984
0000   269 ;               Make changes necessary to support use of mount verification
0000   270 ;               after a connection failure to bring previously online units
0000   271 ;               back online.  This involves eliminating DUTU$FAILOVER which is
0000   272 ;               no longer used, testing CDDB$M_NOCONN as a failover preventer
0000   273 ;               in DUTU$FAILOVER_UCB, setting CDDB$M_NOCONN in DUTU$BUILD_CDDB,
0000   274 ;               and setting CDRP$M_PERM in CDDB_INIT_PRM_CDRP.  Add the
0000   275 ;               following routines; DUTU$TERMINATE_PENDING, DUTU$POST_CDRP,
0000   276 ;               DUTU$DEALLOC_ALL, DUTU$DEALLOC_RSPID_MSG, DUTU$INIT_MSCP_MSG,
0000   277 ;               DUTU$INIT_MSCP_MSG_UNIT, DUTU$RESET_MSCP_MSG, and
0000   278 ;               DUTU$RESTORE_CREDIT.  Change DUTU$FAILOVER_UCB to move entries
0000   279 ;               in the restart queue and to accept the UCB address in R3.
0000   280 ;
0000   281 ;       V03-004 ROW0269         Ralph O. Weber          28-DEC-1983
0000   282 ;               Add DUTU$CREATE_CDDB, a common routine to create, initialize,
0000   283 ;               and link a CDDB.
0000   284 ;
0000   285 ;       V03-003 ROW0262         Ralph O. Weber          24-NOV-1983
```

```
0000   286 ;               Move all UCB lookup and creation to DUTUSUBS.  Cleanup
0000   287 ;               ATTN_MSG processing in DU$IDR.  Implement usage of $DUTUDEF,
0000   288 ;               all device independent UCB fields, and the IOC$GL_DU_CDDB
0000   289 ;               listhead.  This includes addition of the following new
0000   290 ;               routines to this module:
0000   291 ;                > DUTU$POOL_FOR_UNITS - to poll a server for available units.
0000   292 ;                > DUTU$NEW_UNIT - to process unit available attention or get
0000   293 ;                  unit status end messages and create new UCBs where
0000   294 ;                  necessary.
0000   295 ;                > DUTU$SETUP_DUAL_PATH - to check for and where appropriate
0000   296 ;                  setup dual path linkage to a device.
0000   297 ;                > DUTU$LOOKUP_UCB - to locate a UCB on a given CDDB chain
0000   298 ;                  with a given MSCP unit number.
0000   299 ;                > DUTU$INIT_CONN_UCB - to initialize connection dependent
0000   300 ;                  fields in a UCB.
0000   301 ;                > DUTU$FIND_DDB - to locate or create a desired DDB
0000   302 ;                > DUTU$GET_DEVTYPE - to determine VMS device type from MSCP
0000   303 ;                  media identification.
0000   304 ;                > DUTU$GET_DEVNAM - to build a VMS DDCn: device name from
0000   305 ;                  MSCP unit number and MSCP media identification.
0000   306 ;                > DUTU$SETUP_CDP_UCB - to locate and setup local/remote dual
0000   307 ;                  pathed UCBs.
0000   308 ;                > DUTU$SEND_DUPLICATE_UNIT - to send a duplicate unit number
0000   309 ;                  message to the operator.
0000   310 ;                > DUTU$FILL_MSCP_MSG - zero fill an incoming MSCP message.
0000   311 ;
0000   312 ;       V03-002 ROW0261         Ralph O. Weber          22-NOV-1983
0000   313 ;               Build common versions of DUMP_COMMAND (DUTU$DUMP_COMMAND) and
0000   314 ;               DUMP_ENDMESSAGE (DUTU$DUMP_ENDMESSAGE) so that those routines
0000   315 ;               may be removed from the class drivers.  Add the
0000   316 ;               DUTU$SEND_MSCP_MSG routine.
0000   317 ;
0000   318 ;       V03-001 ROW0231         Ralph O. Weber          1-OCT-1983
0000   319 ;               Split failover for a single UCB out of DUTU$FAILOVER into a
0000   320 ;               seperate routine DUTU$FAILOVER_UCB.  This will allow the
0000   321 ;               PACKACK operation to failover a single UCB while attempting to
0000   322 ;               find a path that works to a device.  Also create an
0000   323 ;               independent routine DUTU$SEVER_CDDB to sever CDDB linkages.
0000   324 ;
0000   325 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;
                        $CDDBDEF                        ;Define CDDB offsets
                        $CDRPDEF                        ;Define CDRP offsets
                        $CDTDEF                         ;Define CDT offsets
                        $CRBDEF                         ;Define CRB offsets
                        $DCDEF                          ;Define device classes & types
                        $DDBDEF                         ;Define DDB offsets
                        $DDTDEF                         ;Define DDT offsets
                        $DEVDEF                         ;Define DEVICE CHARACTERISTICS bits
                        $DYNDEF                         ;Define DYN symbols
                        $EMBLTDEF                       ;Define EMB Log Message Types
                        $FKBDEF                         ;Define FKB offsets
                        $IODEF                          ;Define I/O function codes
                        $IPLDEF                         ;Define IPL levels
                        $IRPDEF                         ;Define IRP offsets
                        $MSCPDEF                        ;Define MSCP packet offsets
                        $MSGDEF                         ;Define system message types
                        $MTXDEF                         ;Define MUTEX offsets
                        $PBDEF                          ;Define Path Block offsets
                        $PCBDEF                         ;Define PCB offsets
                        $PDTDEF                         ;Define PDT offsets
                        $PRDEF                          ;Define processor register numbers
                        $RCTDEF                         ;Define RCT offsets
                        $SBDEF                          ;Define System Block offsets
                        $SSDEF                          ;Define System Status values
                        $UCBDEF                         ;Define UCB offsets
                        $VCBDEF                         ;Define VCB offsets
                        $VECDEF                         ;Define VEC offsets

                        $DUTUDEF                        ;Define common class driver CDDB
                                                        ; extensions and other common symbols


                ;
                ; PSECT DEFINITIONS TO LOCATE THE DATA REGION
                ;
00000000                .PSECT  $$$220_DUTU_DATA_00 RD,WRT,EXE,LONG

0000            DUTU$DATA::                             ;Locate base of data region

                ;
                ; MODULE PSECT
                ;
00000000                .PSECT  $$$115_DRIVER LONG

                ;
                ; MACROS:
                ;

                ;++
                ; SWITCH
                ;
                ; Functional description:
                ;
```

```
0000    384 ;       The contents of two data items of the specified length (B,W,L,Q)
0000    385 ;       are exchanged.
0000    386 ;
0000    387 ; Inputs:
0000    388 ;
0000    389 ;       A1      1st argument data item (cannot be R0 or R1)
0000    390 ;       A2      2nd argument data item (cannot be R0 or R1)
0000    391 ;
0000    392 ; Implicit inputs:
0000    393 ;
0000    394 ;       None.
0000    395 ;
0000    396 ; Outputs:
0000    397 ;
0000    398 ;       A1 and A2 are switched.
0000    399 ;
0000    400 ; Implicit outputs:
0000    401 ;
0000    402 ;       R0 is destroyed.  R1 is destroyed if length = Q.
0000    403 ;
0000    404 ;       All other registers are preserved.
0000    405 ;--
0000    406         .MACRO  SWITCH a1, a2, length=L
0000    407         MOV'length'     a1, R0
0000    408         MOV'length'     a2, a1
0000    409         MOV'length'     R0, a2
0000    410         .ENDM   SWITCH
0000    411
0000    412 ;
0000    413 ; EQUATED SYMBOLS:
0000    414 ;
```

```
0000    416          .SBTTL  IRP - CDRP Consistancy Check
0000    417
0000    418
0000    419 ; The following set of ASSUME statements will all be true as long as
0000    420 ;       the IRP and CDRP definitions remain consistent.
0000    421
0000    422          ASSUME  CDRP$L_IOQFL-CDRP$L_IOQFL       EQ      IRP$L_IOQFL
0000    423          ASSUME  CDRP$L_IOQBL-CDRP$L_IOQFL       EQ      IRP$L_IOQBL
0000    424          ASSUME  CDRP$W_IRP_SIZE-CDRP$L_IOQFL    EQ      IRP$W_SIZE
0000    425          ASSUME  CDRP$B_IRP_TYPE-CDRP$L_IOQFL    EQ      IRP$B_TYPE
0000    426          ASSUME  CDRP$B_RMOD-CDRP$L_IOQFL        EQ      IRP$B_RMOD
0000    427          ASSUME  CDRP$L_PID-CDRP$L_IOQFL         EQ      IRP$L_PID
0000    428          ASSUME  CDRP$L_AST-CDRP$L_IOQFL         EQ      IRP$L_AST
0000    429          ASSUME  CDRP$L_ASTPRM-CDRP$L_IOQFL      EQ      IRP$L_ASTPRM
0000    430          ASSUME  CDRP$L_WIND-CDRP$L_IOQFL        EQ      IRP$L_WIND
0000    431          ASSUME  CDRP$L_UCB-CDRP$L_IOQFL         EQ      IRP$L_UCB
0000    432          ASSUME  CDRP$W_FUNC-CDRP$L_IOQFL        EQ      IRP$W_FUNC
0000    433          ASSUME  CDRP$B_EFN-CDRP$L_IOQFL         EQ      IRP$B_EFN
0000    434          ASSUME  CDRP$B_PRI-CDRP$L_IOQFL         EQ      IRP$B_PRI
0000    435          ASSUME  CDRP$L_IOSB-CDRP$L_IOQFL        EQ      IRP$L_IOSB
0000    436          ASSUME  CDRP$W_CHAN-CDRP$L_IOQFL        EQ      IRP$W_CHAN
0000    437          ASSUME  CDRP$W_STS-CDRP$L_IOQFL         EQ      IRP$W_STS
0000    438          ASSUME  CDRP$L_SVAPTE-CDRP$L_IOQFL      EQ      IRP$L_SVAPTE
0000    439          ASSUME  CDRP$W_BOFF-CDRP$L_IOQFL        EQ      IRP$W_BOFF
0000    440          ASSUME  CDRP$L_BCNT-CDRP$L_IOQFL        EQ      IRP$L_BCNT
0000    441          ASSUME  CDRP$W_BCNT-CDRP$L_IOQFL        EQ      IRP$W_BCNT
0000    442          ASSUME  CDRP$L_IOST1-CDRP$L_IOQFL       EQ      IRP$L_IOST1
0000    443          ASSUME  CDRP$L_MEDIA-CDRP$L_IOQFL       EQ      IRP$L_MEDIA
0000    444          ASSUME  CDRP$L_IOST2-CDRP$L_IOQFL       EQ      IRP$L_IOST2
0000    445          ASSUME  CDRP$L_TT_TERM-CDRP$L_IOQFL     EQ      IRP$L_TT_TERM
0000    446          ASSUME  CDRP$B_CARCON-CDRP$L_IOQFL      EQ      IRP$B_CARCON
0000    447          ASSUME  CDRP$Q_NT_PRVMSK-CDRP$L_IOQFL   EQ      IRP$Q_NT_PRVMSK
0000    448          ASSUME  CDRP$L_ABCNT-CDRP$L_IOQFL       EQ      IRP$L_ABCNT
0000    449          ASSUME  CDRP$W_ABCNT-CDRP$L_IOQFL       EQ      IRP$W_ABCNT
0000    450          ASSUME  CDRP$L_OBCNT-CDRP$L_IOQFL       EQ      IRP$L_OBCNT
0000    451          ASSUME  CDRP$W_OBCNT-CDRP$L_IOQFL       EQ      IRP$W_OBCNT
0000    452          ASSUME  CDRP$L_SEGVBN-CDRP$L_IOQFL      EQ      IRP$L_SEGVBN
0000    453          ASSUME  CDRP$L_JNL_SEQNO-CDRP$L_IOQFL   EQ      IRP$L_JNL_SEQNO
0000    454          ASSUME  CDRP$L_DIAGBUF-CDRP$L_IOQFL     EQ      IRP$L_DIAGBUF
0000    455          ASSUME  CDRP$L_SEQNUM-CDRP$L_IOQFL      EQ      IRP$L_SEQNUM
0000    456          ASSUME  CDRP$L_EXTEND-CDRP$L_IOQFL      EQ      IRP$L_EXTEND
0000    457          ASSUME  CDRP$L_ARB-CDRP$L_IOQFL         EQ      IRP$L_ARB
```

```
                                        .SBTTL  ----- INITIALIZATION/REINITIALIZATION ROUTINES -----
                                        .SBTTL  DUTU$CREATE_CDDB - Create and initialize a CDDB
                                ;++
                                ;
                                ; DUTU$CREATE_CDDB - Create and initialize a CDDB
                                ;
                                ; Functional Description:
                                ;
                                ;       This routine allocates pool for a disk or tape class driver CDDB,
                                ;       initializes that CDDB, and links it into the appropriate CDDB chain.
                                ;       Some initialization of the CRB and DDB, previously created by SYSGEN
                                ;       or one of its kin, is also performed.
                                ;
                                ; Inputs:
                                ;
                                ;       R5      UCB address
                                ;
                                ; Implicit Inputs:
                                ;
                                ;       UCB$L_CRB(R5)           CRB address
                                ;       UCB$L_DDB(R5)           DDB address
                                ;       UCB$Q_UNIT_ID(R5)       system ID of remote system housing the server
                                ;                               to which the class driver will connect
                                ;       UCB$L_STS(R5)           if UCB$V_VALID is clear, this UCB does not
                                ;                               represent a real device and should be discarded
                                ;                               when it is no longer needed
                                ;
                                ; Outputs:
                                ;
                                ;       R5      CDDB address
                                ;
                                ;       R0 through R5 are destroyed.
                                ;       All other registers are preserved.
                                ;
                                ; Implicit Outputs:
                                ;
                                ;       The CDDB is completely initialized and ready for use by the class
                                ;       driver.  If UCB$V_VALID in UCB$L_STS is set CDDB$L_ORIGUCB contains
                                ;       the address of the UCB.  Otherwise, CDDB$L_ORIGUCB is zero and the UCB
                                ;       has been deallocated.  See the preamble for DUTU$POLL_FOR_UNITS for a
                                ;       discussion of boot device processing and CDDB$L_ORIGUCB.
                                ;
                                ;       DDB initialization:
                                ;       The DDB is initialized to have no UCBs chained to it.  It is
                                ;       initialized to not be on any CDDB chain.
                                ;
                                ;       CRB initialization:
                                ;       The CRB is threaded onto the TIMELINK chain with an infinite timeout
                                ;       interval.  CRB$L_AUXSTRUC is initialized to point to the CDDB.
                                ;--
                                DUTU$CREATE_CDDB::
                                10$:    POPL    UCB$L_DPC(R5)                   ; Save caller's return addr.
009C C5 8ED0                    MOVZWL  #CDDB$K_DUTULENGTH, R1                  ; Get size of a CDDB.
51   01F8 8F    3C              JSB     G^EXE$ALONONPAGED                       ; Attempt to allocate space.
00000000'GF     16              BLBS    R0, 20$                                 ; Branch if allocation worked.
     08 50      E8
```

```
                                0013   516          FORK_WAIT                               ; Else, wait awhile and
                          EA    0019   517          BRB     10$                             ; try again.
                                001B   518
                                001B   519 20$:     ; R1 - size of allocated block
                                001B   520          ; R2 - address of allocated block
                                001B   521          ; R5 - UCB address
                          26    001B   522          PUSHR   #^M<R1,R2,R5>                   ; Save registers.
                62 51 00  6E  002D   523          MOVC5   #0, (SP), #0, R1, (R2)          ; Zero entire block.
                          38    0030   524          POPR    #^M<R3,R4,R5>                   ; Restore saved registers.
                                0032   525
                                0032   526          WAIT_FOR_IODB                           ; Get write access to the I/O
                                0052   527                                                  ; database.
                          009C C5  DD   0052   528          PUSHL   UCB$L_DPC(R5)           ; Now that waiting is over,
                                0056   529                                                  ; restore caller's return addr.
                                0056   530
                                0056   531          ; R3 - size of allocated block
                                0056   532          ; R4 - address of allocated block
                                0056   533          ; R5 - UCB address
                       08 A4  53  B0   0056   534          MOVW    R3, CDDB$W_SIZE(R4)     ; Setup CDDB size.
                                0054   535          ASSUME  CDDB$B_SUBTYPE  EQ  CDDB$B_TYPE+1
                    0A A4  0164 8F  B0   0054   536          MOVW    #<DYN$C_CLASSDRV -      ; Set type and subtype fields
                                0060   537                    !<DYN$C_CD_CDDB@8>>, -        ; with the CLASSDRV major type
                                0060   538                    CDDB$B_TYPE(R4)               ; and CDDB subtype.
                          53  55  D0   0060   539          MOVL    R5, R3                  ; Move UCB address.
                          55  54  D0   0063   540          MOVL    R4, R5                  ; Move CDDB address.
                                0066   541
                                0066   542          ; R3 - UCB address
                                0066   543          ; R5 - CDDB address
                                0066   544          ASSUME  CDDB$B_SYSTEMID+6  EQ  CDDB$W_STATUS
             0C A5  00CC C3  7D   0066   545          MOVQ    UCB$Q_UNIT_ID(R3), -            ; Record SYSTEM ID.
                                006C   546                    CDDB$B_SYSTEMID(R5)
             12 A5  0084 8F  B0   006C   547          MOVW    #<CDDB$M_INITING -              ; Initialize CDDB status.
                                0072   548                    !CDDB$M_NOCONN>, -
                                0072   549                    CDDB$W_STATUS(R5)
                                0072   550
                                0072   551          ; Set initial controller flages to be used on first Set Controller
                                0072   552          ; Characteristics command.  The flags set are:
                                0072   553          ;    - enable ATTENTION messages
                                0072   554          ;    - enable miscellaneous error log messages
                                0072   555          ;    - enable this host's error log messages
             28 A5  00D0 8F  B0   0072   556          MOVW    #<MSCP$M_CF_ATTN -
                                0078   557                    !MSCP$M_CF_MISC -
                                0078   558                    !MSCP$M_CF_THIS>, -
                                0078   559                    CDDB$W_CNTRLFLGS(R5)
                                0078   560
                          65  65  9E   0078   561          MOVAB   CDDB$L_CDRPQFL(R5), -   ; Initialize Queue listheads.
                                007B   562                    CDDB$L_CDRPQFL(R5)
                    04 A5  65  9E   007B   563          MOVAB   CDDB$L_CDRPQFL(R5), -   ;       "       "       "
                                007F   564                    CDDB$L_CDRPQBL(R5)
                3C A5  3C A5  9E   007F   565          MOVAB   CDDB$L_RSTRTQFL(R5), -  ;       "       "       "
                                0084   566                    CDDB$L_RSTRTQFL(R5)
                40 A5  3C A5  9E   0084   567          MOVAB   CDDB$L_RSTRTQFL(R5), -  ;       "       "       "
                                0089   568                    CDDB$L_RSTRTQBL(R5)
           00B0 C5  00B0 C5  9E   0089   569          MOVAB   CDDB$L_CANCLQFL(R5), -  ;       "       "       "
                                008F   570                    CDDB$L_CANCLQFL(R5)
           00B4 C5  00B0 C5  9E   008F   571          MOVAB   CDDB$L_CANCLQFL(R5), -  ;       "       "       "
                                0096   572                    CDDB$L_CANCLQBL(R5)
```

```
                                 008A   573
                                 008A   574          ASSUME  FKB$B_TYPE EQ <FKB$W_SIZE + 2>  ; Initialize failover fork
                                 008A   575          ASSUME  FKB$B_FIPL EQ <FKB$W_SIZE + 3>  ; block contained in CDDB.
017C C5   08080018 8F    D0     008A   576          MOVL    #<<IPL$_SCS@24> ! <DYN$C_FRK@16> -
                                 0093   577                    ! FKB$K_LENGTH>, -
                                 0093   578                  <FKB$W_SIZE+CDDB$A_2PFKB>(R5)
                                 0093   579
            52   00D0 C5   9E   0093   580          MOVAB   CDDB$A_PRMCDRP(R5), R2          ; Locate permanent CDRP.
                   00BE    30   0098   581          BSBW    CDDB_INIT_PRM_CDRP              ; Initialize it.
            52   0194 C5   9E   009B   582          MOVAB   CDDB$A_DAPCDRP(R5), R2          ; Locate DAP CDRP.
                   00B6    30   00A0   583          BSBW    CDDB_INIT_PRM_CDRP              ; Initialize it.
              54 A5    52   D0   00A3   584          MOVL    R2, CDDB$L_DAPCDRP(R5)          ; Save DAP CDRP address.
                                 00A7   585
           18 A5    24 A3   D0   00A7   586          MOVL    UCB$L_CRB(R3), CDDB$L_CRB(R5)   ; Save CRB address.
                                 00AC   587
                                 00AC   588 ; DDB initialization (and possibly UCB deallocation)
                                 00AC   589
              54    28 A3   D0   00AC   590          MOVL    UCB$L_DDB(R3), R4               ; Get DDB address.
              1C A5    54   D0   00B0   591          MOVL    R4, CDDB$L_DDB(R5)              ; Save it in CDDB.
                    38 A4   D4   00B4   592          CLRL    DDB$L_CONLINK(R4)               ; Remove DDB from any CDDB
                                 00B7   593                                                  ; chain.
                                 00B7   594
           0F 64 A3    0B   E0   00B7   595          BBS     #UCB$V_VALID, -                 ; If UCB is valid, then this
                                 00BC   596                  UCB$L_STS(R3), 45$              ; is the boot device.
                       04 A4   D4   00BC   597          CLRL    DDB$L_UCB(R4)               ; If UCB is not for the boot
                 50      53   D0   00BF   598          MOVL    R3, R0                      ; device, then
                       008A   30   00C2   599          BSBW    DEANONPAGED                 ; unlink UCB from DDB
                         11   11   00C5   600          BRB     50$                         ; and deallocate the UCB.
                                 00C7   601
                                 00C7   602 43$:     BUG_CHECK INCONSTATE, FATAL            ; Non-system disk ORIGUCB.
                                 00CB   603
      53   00000000'8F    D1   00CB   604 45$:     CMPL    #SYS$GL_BOOTUCB, R3             ; Is UCB for the boot device?
                         F3   12   00D2   605          BNEQ    43$                         ; If not, its a fatal error.
              4C A5      53   D0   00D4   606          MOVL    R3, CDDB$L_ORIGUCB(R5)      ; Save boot device UCB addr.
                                 00D8   607
                                 00D8   608 ; Link the new CDDB on the list of all CDDBs attended by this class driver.
                                 00D8   609
                                 00D8   610          ; The new CDDB is linked at the end of a chain of CDDBs for this
                                 00D8   611          ; device type (disk or tape).  While searching for the end of that
                                 00D8   612          ; chain, the system-id of the new CDDB is compared to the system-id
                                 00D8   613          ; of each currently known CDDB.  The new system-id should be unique.
                                 00D8   614          ; Otherwise, the class driver will make a second connection to a
                                 00D8   615          ; remote server with which it is already engaged in transactions.
                                 00D8   616          ; When a match is found, the CDDB, the DDB, CRB, and IDB to which it
                                 00D8   617          ; points will all be unlinked (as appropriate) and deallocated.  All
                                 00D8   618          ; knowledge of the attempt to form two similar connections to the same
                                 00D8   619          ; server will be vaporized.
                                 00D8   620
0000'CF    00000058 8F    C3   00D8   621 50$:     SUBL3   #CDDB$L_CDDBLINK, -             ; Initialize previous CDDB addr.
                         50        00E1   622                  W^<DUTU$DATA + DUTU$L_CDDB_LISTHEAD>, -
                                 00E2   623                  R0
                                 00E2   624
                 51      50   D0   00E2   625 53$:     MOVL    R0, R1                      ; Save previous CDDB address.
              50    58 A1   D0   00E5   626          MOVL    CDDB$L_CDDBLINK(R1), R0     ; Link to next CDDB.
                         10   13   00E9   627          BEQL    59$                         ; Branch if no more CDDBs.
                                 00EB   628          ASSUME  CDDB$S_SYSTEMID EQ 6
           0C A5    0C A0   D1   00EB   629          CMPL    CDDB$B_SYSTEMID(R0), -      ; Is new system-id different
```

```
                                00F0   630                  CDDB$B_SYSTEMID(R5)             ; from one's already in use?
                     F0  12     00F0   631          BNEQ    53$                             ; Branch if different.
           10 A5     10 A0  B1  00F2   632          CMPW    CDDB$B_SYSTEMID+4(R0), -
                                00F7   633                  CDDB$B_SYSTEMID+4(R5)
                     E9  12     00F7   634          BNEQ    53$                             ; Branch if different.
                     1A  11     00F9   635          BRB     DUPLICATE_SYSTEMID              ; Branch if duplicate system-id.
              58 A1  55  D0     00FB   636 59$:      MOVL    R5, CDDB$L_CDDBLINK(R1)         ; Link new CDDB on end of list.
                                00FF   637
                                00FF   638 ; CRB initialization
                                00FF   639
              53  18 A5  D0     00FF   640          MOVL    CDDB$L_CRB(R5), R3              ; Get CRB address.
                  1C A3  D4     0103   641          CLRL    CRB$L_TOUTROUT(R3)              ; Insure access violation
                                0106   642                                                  ; untill a valid timeout
                                0106   643                                                  ; routine is established.
              18 A3  01  CE     0106   644          MNEGL   #1, CRB$L_DUETIME(R3)           ; Set infinite due time.
        00000000'GF  16         010A   645          JSB     G^IOC$THREADCRB                 ; Thread CRB on TIMELINK chain.
              10 A3  55  D0     0110   646          MOVL    R5, CRB$L_AUXSTRUC(R3)          ; Now CDDB is aux structure.
                                0114   647
                     05         0114   648          RSB                                     ; Return to caller.
                                0115   649
                                0115   650 ; Vaporize CDDB with duplicate system-id.
                                0115   651
                                0115   652 DUPLICATE_SYSTEMID:
                                0115   653
              50  1C A5  D0     0115   654          MOVL    CDDB$L_DDB(R5), R0              ; Get DDB address.
                  04 A0  D5     0119   655          TSTL    DDB$L_UCB(R0)                   ; Check for UCBs.  None should
                     37  12     011C   656          BNEQ    OH_NO                           ; be found because this cannot
                     40 A0  D5  011E   657          TSTL    DDB$L_2P_UCB(R0)                ; be the system disk CDDB.
                     32  12     0121   658          BNEQ    OH_NO
        51   34 A0  00000054 8F  C1  0123   659          ADDL3   #<SB$L_DDB - DDB$L_LINK>, -  ; Get starting DDB chain
                                012C   660                  DDB$L_SB(R0), R1                ; address.
                        51  61  D0     012C   661 10$:      MOVL    DDB$L_LINK(R1), R1      ; Link to next DDB.
                            08  13     012F   662          BEQL    20$                     ; Branch if no more DDBs.
                        50  61  D1     0131   663          CMPL    DDB$L_LINK(R1), R0      ; Find DDB which points to
                            F6  12     0134   664          BNEQ    10$                     ; about to vaporize DDB.
                        61  60  D0     0136   665          MOVL    DDB$L_LINK(R0), DDB$L_LINK(R1) ; Unlink vaporizing DDB.
                            14  10     0139   666 20$:      BSBB    DEANONPAGED             ; Deallocate DDB.
                    54  18 A5  D0     013B   667          MOVL    CDDB$L_CRB(R5), R4      ; Get CRB address.
                    50  2C A4  D0     013F   668          MOVL    CRB$L_INTD+VEC$L_IDB(R4), R0 ; Get IDB address.
                            0A  10     0143   669          BSBB    DEANONPAGED             ; Deallocate IDB.
                        50  54  D0     0145   670          MOVL    R4, R0                  ; Setup CRB.
                            05  10     0148   671          BSBB    DEANONPAGED             ; Deallocate CRB.
                            8E  D5     014A   672          TSTL    (SP)+                   ; Pop caller's return address.
                        50  55  D0     014C   673          MOVL    R5, R0                  ; Setup CDDB.
                                014F   674 DEANONPAGED:
                00000000'GF 17         014F   675          JMP     G^EXE$DEANONPAGED       ; Deallocate it and terminate
                                0155   676                                                  ; this fork thread.
                                0155   677
                                0155   678 OH_NO:   BUG_CHECK MSCPCLASS, FATAL             ; Found duplicate system-id on
                                0159   679                                                  ; a CDDB with units already
                                0159   680                                                  ; associated.
```

```
                     0159   682 ;++
                     0159   683 ;
                     0159   684 ; CDDB_INIT_PRM_CDRP - Initialize a CDRP permanently attached to a CDDB
                     0159   685 ;
                     0159   686 ; Functional Description:
                     0159   687 ;
                     0159   688 ;       This routine initializes one of the CDRPs permanently attached to a
                     0159   689 ;       CDDB.  Because the CDRP is included in the CDDB allocation, it is
                     0159   690 ;       zeroed when the CDDB is zeroed.  Therefore, only interesting, non-
                     0159   691 ;       zero fields are initialized here.
                     0159   692 ;
                     0159   693 ; Inputs:
                     0159   694 ;
                     0159   695 ;       R2 - CDRP address
                     0159   696 ;       R5 - CDDB address
                     0159   697 ;
                     0159   698 ; Outputs:
                     0159   699 ;
                     0159   700 ;       R0 & R1 destroyed.
                     0159   701 ;       All other registers are preserved.
                     0159   702 ;--
                     0159   703
                     0159   704 CDDB_INIT_PRM_CDRP:
                     0159   705
                     0159   706         ASSUME  CDDB$K_DUTULENGTH LT 32758
          51  55  52  C3   0159   707         SUBL3   R2, R5, R1                      ; Compute offset from CDRP
                     015D   708                                                     ; base to CDDB base.
           08 A2  51  B0   015D   709         MOVW    R1, CDRP$W_CDRPSIZE(R2)         ; Store offset in CDRP size.
                     0161   710         ASSUME  CDRP$B_FIPL EQ CDRP$B_CD_TYPE+1
 0A A2  0839 8F  B0   0161   711         MOVW    #<DYN$C_CDRP + <IPL$_SCS@8>>, -  ; Initialize type and fork
                     0167   712                 CDRP$B_CD_TYPE(R2)              ; IPL.
        28 A2  D4   0167   713         CLRL    CDRP$L_RWCPTR(R2)               ; Clear pointer to RWAITCNT.
     40 A2  08  D0   016A   714         MOVL    #CDRP$M_PERM, -                 ; Flag this a perm. IRP/CDRP.
                     016E   715                 CDRP$L_DUTUFLAGS(R2)
                  05 016E   716         RSB
```

```
                        .SBTTL  DUTU$POLL_FOR_UNITS - Poll a server for its available units
                ;++
                ;
                ; DUTU$POLL_FOR_UNITS - Poll a server for its available units
                ;
                ; Functional Description:
                ;
                ;       This routine polls a MSCP server to determin what units it has
                ;       available.  This is done by sending GET UNIT STATUS commands with the
                ;       NEXT UNIT modifier until all known units have been reported.
                ;       Functional units which are not represented in the I/O data base have a
                ;       UCB added.
                ;
                ; Special Processing for the Boot Device:
                ;
                ;       When the class driver is controlling the boot device, DUTU$CREATE_CDDB
                ;       detects a UCB which is valid (UCB$V_VALID set in UCB$L_STS).  When
                ;       this is the case, the UCB address is copied to CDDB$L_ORIGUCB.  In all
                ;       other cases, CDDB$L_ORIGUCB is left zero and the UCB passed to the
                ;       driver's controller initialization routine is deallocated.
                ;
                ;       While polling for units, this routine should find the boot device.  It
                ;       will match the unit number to the unit number in the UCB pointed to by
                ;       CDDB$L_ORIGUCB and when a match occurs it will begin special boot
                ;       device processing.  After the special boot device processing is
                ;       initiated for the first time, CDDB$L_ORIGUCB will be cleared.  This
                ;       makes the special handling of the boot device a once per system boot
                ;       occurence.
                ;
                ;       This processing differs from other poll units processing in two ways.
                ;       The original UCB -- handed the class driver during controller
                ;       initialization -- is used, not a newly created UCB.  Also, a special,
                ;       limited form of mount verification (controlled completely by this
                ;       routine and its subroutines) is performed.  This limited mount
                ;       verification only performs a PACKACK function.  (As opposed to regular
                ;       mount verification which performs both a PACKACK and a volume
                ;       validation operation).  There is no memory stored volume information
                ;       against which to validate the boot volume.  Also, system requirements
                ;       at this stage of the booting process are limited to the boot device
                ;       being accessible to the boot procedures (i.e. ONLINE).
                ;
                ;       N.B. this method works equally well for disks or tapes.  Although
                ;       booting from tapes is not supported, its not prohibited by this code.
                ;
                ; Inputs:
                ;
                ;       R3      CDDB address
                ;       R4      PDT address
                ;       R5      address of the DAP CDRP
                ;
                ; Implicit Inputs:
                ;
                ;       CDDB$L_ORIGUCB(R3)      boot device UCB address, if first init. or
                ;                               boot device by class driver; otherwise zero
                ;       CDRP$L_RSPID(R5)        a valid RSPID
                ;       CDRP$L_MSG_BUF(R5)      address of a valid SCS message buffer
                ;       CDRP$W_SIZE(R5)         offset to the CDDB
```

```
                     016F    775 ;       CDDB$W_STATUS(R3)       CDDB$V_DAPBSY set
                     016F    776 ;
                     016F    777 ;       The normal class driver MSCP operation timeout mechanism must be
                     016F    778 ;       enabled.
                     016F    779 ;
                     016F    780 ; Outputs:
                     016F    781 ;
                     016F    782 ;       R3      CDDB address (same as input)
                     016F    783 ;       R4      PDT address  (same as input)
                     016F    784 ;       R5      CDRP address (same as input)
                     016F    785 ;
                     016F    786 ;       R0 through R5 are destroyed.
                     016F    787 ;       All other registers are preserved.
                     016F    788 ;--
                     016F    789
                     016F    790         .ENABLE LSB
                     016F    791
                     016F    792 DUTU$POLL_FOR_UNITS::
                     016F    793
     44 A3 8ED0      016F    794         POPL    CDDB$L_SAVED_PC(R3)     ; Save caller's return in CDDB field.
12 A3   20   A8      0173    795         BISW    #CDDB$M_POLLING, -      ; Set bit indicating polling in
                     0177    796                 CDDB$W_STATUS(R3)       ; progress.
                     0177    797         ; Clearing CDRP$L_MEDIA causes the search for units to begin with
                     0177    798         ; unit 1.  MSCP dictates that unit 0 is the last unit to be polled.
     D8 A5   D4      0177    799         CLRL    CDRP$L_MEDIA(R5)
                     017A    800         ALLOC_RSPID                     ; Allocate a response-id.
                     0180    801         ALLOC_MSG_BUF                   ; Allocate a message buffer.
     50 50   E9      0183    802         BLBC    R0, 139$                ; Branch if connection broken already.
                     0186    803
                     0186    804 POLL_LOOP:
                     0186    805
                     0186    806         INIT_MSCP_MSG                   ; Initialize buffer for MSCP message.
                     0189    807
        D8 A5   B6   0189    808         INCW    CDRP$L_MEDIA(R5)        ; Proceed with next unit.
  04 A2 D8 A5   B0   018C    809         MOVW    CDRP$L_MEDIA(R5), -     ; Put next unit number in MSCP
                     0191    810                 MSCP$W_UNIT(R2)         ; command.
                     0191    811
  08 A2   03   90    0191    812         MOVB    #MSCP$K_OP_GTUNT, -     ; Command is GET UNIT STATUS.
                     0195    813                 MSCP$B_OPCODE(R2)
  0A A2   01   B0    0195    814         MOVW    #MSCP$M_MD_NXUNT, -     ; Modifier is NEXT UNIT.
                     0199    815                 MSCP$W_MODIFIER(R2)
                     0199    816
                     0199    817         SEND_MSCP_MSG DRIVER            ; Returns with END PACKET addr. in R2.
                     019C    818
D8 A5   04 A2   3C   019C    819         MOVZWL  MSCP$W_UNIT(R2), -      ; Save unit number located.
                     01A1    820                 CDRP$L_MEDIA(R5)
                     01A1    821
                     01A1    822         ; Check for having located the original UCB, possibly the system
                     01A1    823         ; device, presented to the driver's controller initialization routine.
                     01A1    824
     50   4C A3   D0 01A1    825         MOVL    CDDB$L_ORIGUCB(R3), R0  ; Get original UCB address.
             08   13 01A5    826         BEQL    30$                     ; Branch if no such UCB exists.
00D4 C0   04 A2   B1 01A7    827         CMPW    MSCP$W_UNIT(R2), -      ; Does UCB unit match end-message
                     01AD    828                 UCB$W_MSCPUNIT(R0)      ; unit number?
             25   13 01AD    829         BEQL    135$                    ; If match, go to special case code.
                     01AF    830
                     01AF    831 30$:    ; Check end-message -- determine whether or not we want the unit it
```

```
                                01AF    832          ; describes in our I/O data base.  We want the unit if the status is
                                01AF    833          ; one of SUCC, AVLBL, DRIVE, or OFFLN (NOVOL).
                                01AF    834
                                01AF    835          ASSUME  MSCP$V_ST_MASK EQ 0
        50   0A A2   FFE0 8F    AB  01AF    836          BICW3   #^cMSCP$M_ST_MASK, -    ; Extract major MSCP status.
                                01B6    837                  MSCP$W_STATUS(R2), R0
                                01B6    838          DISPATCH R0, type=W, prefix=MSCP$K_ST_, < -
                                01B6    839                  <SUCC,40$>, -
                                01B6    840                  <AVLBL,40$>, -
                                01B6    841                  <DRIVE,40$>, -
                                01B6    842                  <OFFLN,35$> -
                                01B6    843                  >
                          2E    11  01D2    844          BRB     90$                     ; If none of the above, ignore message.
                          5C    11  01D4    845 135$:    BRB     DO_ORIG_UCB             ; Branch assist.
                          48    11  01D6    846 139$:    BRB     POLL_CONN_BROKE         ; Branch assist.
                                01D8    847
                                01D8    848 35$:     ; For offline devices, accept only those who have no volume mounted or
                                01D8    849          ; are disabled via the RUN/STOP switch, subcode NOVOL
          25 0A A2    05    E1  01D8    850          BBC     #MSCP$V_SC_NOVOL, -     ; Is the subcode NOVOL?
                                01DD    851                  MSCP$W_STATUS(R2), 90$  ; Branch if not NOVOL.
                                01DD    852
             51   46 A5    3C  01DD    853 40$:     MOVZWL  CDRP$W_ENDMSGSIZ(R5), R1; Restore MSCP message size.
                  025D      30  01E1    854          BSBW    DUTU$NEW_UNIT           ; Make sure a UCB exists for this unit.
                    52      D5  01E4    855          TSTL    R2                      ; Does UCB exist?
                    3B      13  01E6    856          BEQL    RESTART_POLL            ; Branch if UCB does not exist.
                                01E8    857
                                01E8    858 FINISH_ORIGUCB:
             50   1C A5    D0  01E8    859          MOVL    CDRP$L_MSG_BUF(R5), R0  ; Get message buffer address.
                                01EC    860          ASSUME  MSCP$K_ST_SUCC EQ 0     ; If the unit found was online,
          0A A0    1F    B3  01EC    861          BITW    #MSCP$M_ST_MASK, -      ; the unit flags are valid.
                                01F0    862                  MSCP$W_STATUS(R0)
                    06    12  01F0    863          BNEQ    60$                     ; Branch if unit not online.
    00E0 C2   0E A0    B0  01F2    864          MOVW    MSCP$W_UNT_FLGS(R0), -  ; If online, copy unit flags to UCB.
                                01F8    865                  UCB$W_UNIT_FLAGS(R2)
                                01F8    866
                                01F8    867 60$:     ; Insure that permanent CDRP and DAP CDRP have a UCB address.
      008C C3     52    D0  01F8    868          MOVL    R2, CDDB$L_PRMUCB(R3)   ; Set permanent CDRP UCB address.
      0150 C3     52    D0  01FD    869          MOVL    R2, CDDB$L_DAPUCB(R3)   ; Set DAP CDRP UCB address.
                              0202    870
             D8 A5    B5  0202    871 90$:     TSTW    CDRP$L_MEDIA(R5)        ; Was unit found zero?
                0F    13  0205    872          BEQL    95$                     ; If zero, all done:  exit.
                          0207    873          RECYCH_MSG_BUF                  ; Recycle message buffer.
             13 50    E9  020A    874          BLBC    R0, POLL_CONN_BROKE     ; Branch if connection is broken.
                          020D    875          RECYCL_RSPID                    ; Recycle response-id.
              FF70    31  0213    876          BRW     POLL_LOOP               ; Loop till unit zero found.
                          0216    877
              0AAF    30  0216    878 95$:     BSBW    DUTU$DEALLOC_ALL        ; Release all CDRP resources.
          12 A3   20    AA  0219    879          BICW    #CDDB$M_POLLING, -      ; Indicate that polling no longer is
                          021D    880                  CDDB$W_STATUS(R3)       ; in progress.
             44 B3    17  021D    881          JMP     @CDDB$L_SAVED_PC(R3)    ; Return to original caller.
                          0220    882
                          0220    883 ; The connection has broken during a polling operation.  Release CDRP
                          0220    884 ; resources and abort the polling thread.
                          0220    885
                          0220    886 POLL_CONN_BROKE:
              0AA5    31  0220    887          BRW     DUTU$DEALLOC_ALL        ; Release all CDRP resources and
                          0223    888                                          ; terminate the polling fork thread.
```

```
                     0223   889
                     0223   890 ; At this point, something went wrong trying to build a UCB for a unit
                     0223   891 ; discovered in the polling process.  Most likely, there was not enough
                     0223   892 ; non-paged pool to accomidate the UCB.  The process of polling for units
                     0223   893 ; will be restarted from the beginning.  Hopefully, by the time we return
                     0223   894 ; to the unit which could not be configured, some non-paged pool will be
                     0223   895 ; available to configure it.  Otherwise, this is an infinite loop.
                     0223   896
                     0223   897 RESTART_POLL:
                     0223   898         RECYCL_MSG_BUF                  ; Recycle MSCP message buffer.
                     0226   899         RECYCL_RSPID                    ; Recycle the response ID.
D8 A5   D4   022C    900         CLRL    CDRP$L_MEDIA(R5)                ; Restart polling at unit 1.
  FF54   31  022F    901         BRW     POLL_LOOP                       ; Restart loop.
```

```
                                0232   903 ;
                                0232   904 ; HAND PROCESS BOOT DEVICE UCB
                                0232   905 ;
                                0232   906 ;       Boot device UCB fields are initialized from the get unit status
                                0232   907 ;       packet.  A limited form of mount verification processing -- involving
                                0232   908 ;       only a IO$_PACKACK -- is begun.  Finally, the UCB is properly linked
                                0232   909 ;       into the I/O database.
                                0232   910 ;
                                0232   911 ; Inputs:
                                0232   912 ;               R0      UCB address
                                0232   913 ;               R2      MSCP message buffer address
                                0232   914 ;               R3      CDDB address
                                0232   915 ;               R4      PDT address
                                0232   916 ;               R5      CDRP address
                                0232   917 ;
                                0232   918 ; Outputs:
                                0232   919 ;
                                0232   920 ;               R0 & R1 scratch
                                0232   921 ;               R2      UCB address
                                0232   922 ;               R3      CDDB address (unchanged)
                                0232   923 ;               R4      PDT address (unchanged)
                                0232   924 ;               R5      CDRP address (unchanged)
                                0232   925 ;
                                0232   926
                                0232   927 DO_ORIG_UCB:
                     39   BB    0232   928         PUSHR   #^M<R0,R3,R4,R5>         ; Save registers.
           51    46 A5   3C    0234   929         MOVZWL  CDRP$W_ENDMSGSIZ(R5), R1; Restore length of MSCP message.
                 0C72    30    0238   930         BSBW    DUTU$FILL_MSCP_MSG       ; Zero fill the MSCP message.
      008C C0    1C A2   D0    023B   931         MOVL    MSCP$L_MEDIA_ID(R2), -   ; Copy MSCP media identification.
                                0241   932                 UCB$L_MEDIA_ID(R0)
      00D4 C0    04 A2   B0    0241   933         MOVW    MSCP$W_UNIT(R2), -       ; Copy MSCP unit number.
                                0247   934                 UCB$W_MSCPUNIT(R0)
                55    50   D0    0247   935         MOVL    R0, R5                   ; Setup UCB address.
                00000000'GF   16    024A   936         JSB     G^IOC$SEVER_UCB          ; Unlink UCB.
                                0250   937         CREATE_FORK -                    ; Create a fork thread to relink
                                0250   938                 LINK_NEW_UCB, -          ; the original UCB.
                                0250   939                 frkblk = (R5), -
                                0250   940                 mask = <^M<>>
                55    6E    D0    0257   941         MOVL    (SP), R5                 ; Restore UCB address.
           58 64 A5   09    E0    025A   942         BBS     #UCB$V_MSCP_INITING, -   ; Branch if UCB still initing;
                                025F   943                 UCB$W_STS(R5), 388$      ; it should not be!!!
                                025F   944
                50    0088 C5   D0    025F   945         MOVL    UCB$L_DDT(R5), R0        ; Get DDT address.
        20 A0    00000000'8F   D1    0264   946         CMPL    #IOC$RETURN, -           ; Does this driver do mount
                                026C   947                 DDT$L_MNTVER(R0)         ; verification?
                      41   13    026C   948         BEQL    320$                     ; If not, skip simulated mnt. ver.
                51    00C4 8F   3C    026E   949         MOVZWL  #IRP$K_LENGTH, R1        ; Get IRP size.
                 00000000'GF   16    0273   950         JSB     G^EXE$ALONONPAGED        ; Allocate one.
                        3F   50    E9    0279   951         BLBC    R0, 399$                 ; Branch if error.
                             26    BB    027C   952         PUSHR   #^M<R1,R2,R5>            ; Save valuable registers.
    62    51    00    6E    00    2C    027E   953         MOVC5   #0, (SP), #0, R1, (R2)   ; Zero the entire packet.
                             2C    BA    0284   954         POPR    #^M<R2,R3,R5>            ; Restore saved registers.
                   08 A3   52    B0    0286   955         MOVW    R2, IRP$W_SIZE(R3)       ; Set packet size.
                   0A A3   0A    90    028A   956         MOVB    #DYN$C_IRP, -            ; Set packet type.
                                028E   957                 IRP$B_TYPE(R3)
                                028E   958         ASSUME  IO$_PHYSICAL GE IO$_PACKACK
                   20 A3   08    B0    028E   959         MOVW    #IO$_PACKACK, -          ; Set function code.
```

```
                        0292   960                  IRP$W_FUNC(R3)
2A A3   2100 8F   B0  0292   961          MOVW    #<IRP$M_PHYSIO -          ; Set IRP status flags.
                      0298   962                    !IRP$M_MVIRP>, IRP$W_STS(R3)
 03 68 A5    08   E2  0298   963          BBSS    #UCB$V_MSCP_MNTVERIP, -   ; Mark the UCB mount ver. in
                      029D   964                  UCB$W_DEVSTS(R5), 310$    ; progress, and bump wait
          56 A5   B6  029D   965          INCW    UCB$W_RWAITCNT(R5)        ; if that's needed.
       1C A3   55   D0  02A0   966 310$:   MOVL    R5, IRP$L_UCB(R3)         ; Set UCB address in IRP.
    0C A3   BF'AF   9E  02A4   967          MOVAB   B^END_ORIG_UCB_IO, -      ; Set completion routine in IRP.
                      02A9   968                  IRP$L_PID(R3)
     00000000'GF    16  02A9   969          JSB     G^IOC$INITIATE            ; Begin the request.
                      02AF   970
             3C   BA  02AF   971 320$:   POPR    #^M<R2,R3,R4,R5>          ; Restore registers.
          4C A3   D4  02B1   972          CLRL    CDDB$L_ORIGUCB(R3)        ; Indicate no more original UCB.
           FF31   31  02B4   973          BRW     FINISH_ORIGUCB            ; Rejoin polling loop.
                      02B7   974
                      02B7   975 388$:   BUG_CHECK MSCPCLASS, FATAL         ; Boot UCB initialization waited for
                      02BB   976                                            ; something.  This should never happen.
                      02BB   977
                      02BB   978 399$:   BUG_CHECK MSCPCLASS, FATAL         ; Could not allocate IRP for boot
                      02BF   979                                            ; device PACKACK.  We could recover
                      02BF   980                                            ; from this, but it should never happen.
                      02BF   981
                      02BF   982          .DISABLE LSB
```

```
                     02BF    984 ;
                     02BF    985 ; I/O POST ROUTINE FOR ORIG_UCB IO$_PACKACK
                     02BF    986 ;
                     02BF    987 ;       If the packack was successful, completion of mount verification is
                     02BF    988 ;       simulated.  Otherwise, the packack is resubmitted.
                     02BF    989 ;
                     02BF    990 ; Inputs:
                     02BF    991 ;
                     02BF    992 ;       R5      IRP address
                     02BF    993 ;       IPL     IPL$_IOPOST
                     02BF    994 ;
                     02BF    995 ; Outputs:
                     02BF    996 ;
                     02BF    997 ;       R0 - R5 destroyed.
                     02BF    998 ;       All other registers and IPL preserved.
                     02BF    999 ;
                     02BF   1000
                     02BF   1001 END_ORIG_UCB_IO:
                     02BF   1002
      1F 38 A5   E9  02BF   1003         BLBC    IRP$L_IOST1(R5), 90$            ; Branch if PACKACK error.
         50   55   D0  02C3   1004         MOVL    R5, R0                          ; Copy IRP address.
   55    1C A0   D0  02C6   1005         MOVL    IRP$L_UCB(R0), R5               ; Get UCB address.
         FE82    30  02CA   1006         BSBW    DEANONPAGED                     ; Deallocate IRP.
                 02CD   1007         DSBINT  UCB$B_FIPL(R5)                  ; Raise to fork IPL.
            53   D4  02D4   1008         CLRL    R3                              ; Signal end mount ver.
50    0088 C5   D0  02D6   1009         MOVL    UCB$L_DDT(R5), R0               ; Get DDT address.
      20 B0   16  02DB   1010         JSB     @DDT$L_MNTVER(R0)               ; Call end mount ver. rout.
                 02DE   1011         ENBINT                                  ; Restore IPL.
                 05  02E1   1012         RSB                                     ; Return to our caller.
                 02E2   1013
   53   55   D0  02E2   1014 90$:    MOVL    R5, R3                          ; Copy IRP address.
55    60 A3   9E  02E5   1015         MOVAB   IRP$L_FQFL(R3), R5              ; Setup CDRP for wait.
                 02E9   1016         FORK_WAIT                               ; Wait a while.
55    1C A3   D0  02EF   1017         MOVL    IRP$L_UCB(R3), R5               ; Get UCB address.
00000000'GF   17  02F3   1018         JMP     G^IOC$INITIATE                  ; Try PACKACK again.
```

```
                                .SBTTL  DUTU$UNITINIT - Class driver unit initialization routine
                        ;++
                        ;
                        ; DUTU$UNITINIT - Class driver unit initialization routine
                        ;
                        ; Functional Description:
                        ;
                        ;       The operating system calls this routine after calling the controller
                        ;       initialization routine:
                        ;
                        ;               at system startup
                        ;               during driver loading
                        ;               during recovery from a power failure
                        ;
                        ;       For the class drivers, the primary purpose of this routine is deletion
                        ;       of unnecessary UCBs created by SYSGEN.  For the purposes of determining
                        ;       whether or not to delete a UCB that UCB can be in one of three states
                        ;       upon entry to this routine:
                        ;
                        ;           1.  The UCB can be online (i.e. UCB$V_ONLINE is set in UCB$W_STS).
                        ;               In this case, the UCB should not be deleted.  The UCB is
                        ;               either the system disk UCB, supplied at startup, or it is a
                        ;               UCB built during normal system operation by the class driver
                        ;               and this call was caused by a power failure.
                        ;
                        ;           2.  The UCB can be offline after a power failure (i.e.
                        ;               UCB$V_ONLINE is clear and UCB$V_POWER is set in UCB$W_STS).
                        ;               Again, the UCB should not be deleted.  This call is the result
                        ;               of a power failure and the UCB is one of the CDP UCBs for a
                        ;               local device which has been found to be accessible via a class
                        ;               driver path.
                        ;
                        ;           3.  The UCB can be offline with no power failure having occured
                        ;               (i.e. both UCB$V_ONLINE and UCB$V_POWER are clear in UCB$W_STS).
                        ;               This UCB should be removed from the DDB chain and deleted.  It
                        ;               is a bogus UCB supplied by SYSGEN.
                        ;
                        ;       N.B. These tests depend heavily on two facts:
                        ;
                        ;           a.  The DPT_STORE initialization macros do NOT set UCB$V_ONLINE.
                        ;
                        ;           b.  The class driver controller initialization does set
                        ;               UCB$V_ONLINE in any UCB that potentially represents the system
                        ;               device.
                        ;
                        ;       To make life simpler and to keep SDA device desplays from looking
                        ;       "funny," this routine always clears UCB$V_POWER.  No other part of the
                        ;       class drivers test, reset, or care about this bit, but leaving it on
                        ;       eternally after a power failure will look wierd.
                        ;
                        ; Inputs:
                        ;
                        ;       R5      UCB address
                        ;
                        ; Outputs:
                        ;
                        ;       R0 and R1 are destroyed.
```

```
                        02F9  1077 ;       All other registers are preserved.
                        02F9  1078 ;--
                        02F9  1079
                        02F9  1080 DUTU$UNITINIT::
                        02F9  1081
 0F 64 A5    05    E4   02F9  1082          BBSC    #UCB$V_POWER, UCB$W_STS(R5), -  ; Clear the power failure bit
                        02FE  1083                  90$                             ; and branch if it was set.
 0A 64 A5    04    E0   02FE  1084          BBS     #UCB$V_ONLINE, UCB$W_STS(R5), - ; Branch if the online bit
                        0303  1085                  90$                             ; is set.
                        0303  1086
                        0303  1087          ASSUME  UCB$V_DELETEUCB GE 16
   66 A5    01    88   0303  1088          BISB    #<UCB$M_DELETEUCB @ -16>, -     ; Else, set the delete this
                        0307  1089                  UCB$L_STS+2(R5)                 ; UCB bit.
 00000000'GF    16    0307  1090          JSB     G^IOC$DELETE_UCB                ; Then, delete this UCB.
                        030D  1091
                  05    030D  1092 90$:     RSB                                     ; Return to caller.
```

```
                        .SBTTL  DUTU$FAILOVER_UCB - Failover a single UCB
                ;+
                ; DUTU$FAILOVER_UCB - Failover a single UCB
                ;
                ; Functional Description:
                ;
                ;       If conditions permit, the UCB whose address is in R5 has its primary
                ;       path switched to its secondary path and vise versa.  The secondary
                ;       path CDDB must exist and have all of the following bits clear for
                ;       failover to proceed:
                ;
                ;               CDDB$W_STATUS bit       meaning (when clear)
                ;
                ;               CDDB$V_SNGLSTRM         not in single stream mode
                ;               CDDB$V_INITING          not initializing
                ;               CDDB$V_RESYNCH          not resynchronizing
                ;               CDDB$V_POLLING          not polling for units
                ;               CDDB$V_NOCONN           has a good connection
                ;
                ;       The failover attempt is also prohibited whenever the UCB to be failed
                ;       over has incomplete cancel operations.
                ;
                ; Inputs:
                ;
                ;       R3      Address of UCB to be failed-over
                ;
                ; Implicit Inputs:
                ;
                ;       UCB$L_CDDB(R3)          Address of current primary CDDB
                ;       UCB$L_2P_CDDB(R3)       Address of current secondary CDDB
                ;       UCB$L_DDB(R3)           Address of current primary DDB
                ;       UCB$L_2P_DDB(R3)        Address of current secondary DDB
                ;       UCB$L_LINK(R3)          Address of next UCB in primary DDB chain
                ;       UCB$L_2P_LINK(R3)       Address of next UCB in secondary DDB chain
                ;       UCB$L_CDDB_LINK(R3)     Address of next UCB in CDDB chain
                ;       UCB$W_MSCPUNIT(R3)      MSCP unit number for UCB
                ;       UCB$W_UNIT(R3)          Unit number for UCB
                ;       UCB$W_DEVSTS(R3)        UCB$V_MSCP_WAITBMP set if current (old)
                ;                               connection is performing a reconnect
                ;
                ; Outputs:
                ;
                ;       R0      SS$_NORMAL ==> failover succesfully performed
                ;               SS$_MEDOFL ==> failover could not be performed
                ;
                ;       All other registers preserved
                ;
                ; Implicit Outputs:
                ;
                ;       UCB unlinked from primary path, primary and secondary paths switched,
                ;       and UCB linked into former secondary path which is no primary.  UCB
                ;       fields altered by this operation:
                ;
                ;       UCB$L_CDDB(R3)          Address of new primary CDDB
                ;       UCB$L_2P_CDDB(R3)       Address of new secondary CDDB
                ;       UCB$L_DDB(R3)           Address of new primary DDB
                ;       UCB$L_2P_DDB(R3)        Address of new secondary DDB
```

```
                     030E  1151 ;       UCB$L_LINK(R3)          Address of next UCB in primary DDB chain
                     030E  1152 ;       UCB$L_2P_LINK(R3)       Address of next UCB in CDDB chain
                     030E  1153 ;       UCB$L_CRB(R3)           Address of CRB for new primary CDDB
                     030E  1154 ;       UCB$W_RWAITCNT(R3)      bumped if new CDDB has CDDB$V_RECONNECT set:
                     030E  1155 ;                               decremented if new CDDB has CDDB$V_RECONNECT
                     030E  1156 ;                               clear and UCB$V_MSCP_WAITBMP is set
                     030E  1157 ;       UCB$W_DEVSTS(R3)        UCB$V_WAITBMP is made to match CDDB$V_RECONNECT
                     030E  1158 ;                               in the new CDDB
                     030E  1159 ;
                     030E  1160 ;       Then the original primary CDDB is restart queue scanned for CDRPs with
                     030E  1161 ;       CDRP$L_UCB matching the input UCB.  Each CDRP located is placed --
                     030E  1162 ;       in the order found -- at the head of the UCB's I/O queue.
                     030E  1163 ;
                     030E  1164 ; Side Effects:
                     030E  1165 ;
                     030E  1166 ;       Because the I/O database DDB chains are updated to reflect a
                     030E  1167 ;       successful failover operation in a separate fork thread, that change
                     030E  1168 ;       may not appear immediately after this routine is called.  However, all
                     030E  1169 ;       changes required to successfully access the device have been made upon
                     030E  1170 ;       exit from this routine and I/O request processing can proceed
                     030E  1171 ;       immediately.  The I/O database will be updated as soon as the separate
                     030E  1172 ;       fork thread detects that the I/O database mutex is unowned.
                     030E  1173 ;-
                     030E  1174
                     030E  1175 DUTU$FAILOVER_UCB::
                     030E  1176
          3E   BB    030E  1177          PUSHR   #^M<R1,R2,R3,R4,R5>             ; Save registers.
1A 3C A3  04   E1    0310  1178          BBC     #DEV$V_2P, -                    ; Branch if device not dual-
                     0315  1179                  UCB$L_DEVCHAR2(R3), 90$         ; path configured.
       55  53   D0    0315  1180          MOVL    R3, R5                          ; Setup UCB address.
    53 00C0  C5   D0    0318  1181          MOVL    UCB$L_2P_CDDB(R5), R3           ; Get secondary CDDB.
          10   13    031D  1182          BEQL    90$                             ; Branch if no secondary CDDB.
12 A3 00B5  8F   B3    031F  1183          BITW    #<CDDB$M_SNGLSTRM -             ; Is secondary CDDB ready for
                     0325  1184                   !CDDB$M_INITING -              ; failover?
                     0325  1185                   !CDDB$M_RESYNCH -
                     0325  1186                   !CDDB$M_POLLING -
                     0325  1187                   !CDDB$M_NOCONN>, -
                     0325  1188                  CDDB$W_STATUS(R3)
          08   12    0325  1189          BNEQ    90$                             ; Branch if sec. CDDB not ready.
          54   D4    0327  1190          CLRL    R4                              ; Signal test UCB only.
      0675   30    0329  1191          BSBW    DUTU$CHECK_NOCANCEL             ; Check for active cancels.
    08 50   E8    032C  1192          BLBS    R0, 100$                        ; Branch if no active cancels.
 50 01A4 8F   3C    032F  1193 90$:     MOVZWL  #SS$_MEDOFL, R0                 ; Else, indicate failure.
      0073   31    0334  1194          BRW     580$                            ; Branch to routine exit.
                     0337  1195
                     0337  1196
      041C   30    0337  1197 100$:    BSBW    DUTU$SEVER_CDDB                 ; Unlink UCB from failed CDDB
                     033A  1198                                                  ; UCBs chain.
                     033A  1199          SWITCH  UCB$L_CDDB(R5), -               ; Switch primary and secondary
                     033A  1200                  UCB$L_2P_CDDB(R5)               ; CDDB pointers.
      04D7   30    034B  1201          BSBW    DUTU$INIT_CONN_UCB              ; Adjust connection dependent
                     034E  1202                                                  ; UCB fields.
                     034E  1203          ; Adjust UCB$V_MSCP_WAITBMP and UCB$W_RWAITCNT to reflect any change
                     034E  1204          ; in connection state between the former and the current connection.
                     034E  1205          ; The following table describes the needed adjustment:
                     034E  1206          ;
                     034E  1207          ; WAITBMP  RWAITCNT     RECONNECT       WAITBMP  RWAITCNT  RWAITCNT
```

```
                     034E  1208          ;   old      old        new CDDB          new      new    adjustment
                     034E  1209          ;
                     034E  1210          ;    0        0          0                0        0         0
                     034E  1211          ;    0        0          1                1       +1        +1
                     034E  1212          ;    1       +1          0                0        0        -1
                     034E  1213          ;    1       +1          1                1       +1         0
                     034E  1214          ;
                     034E  1215          ; Start by assuming new CDDB$V_RECONNECT is set and backout the
                     034E  1216          ; adjustments if it is not.
                     034E  1217
03 68 A5   0A   E2   034E  1218          BBSS    #UCB$V_MSCP_WAITBMP, -          ; Assume that a reconnect is in
                     0353  1219                  UCB$W_DEVSTS(R5), 125$          ; progress and alter wait count
       56 A5   B6    0353  1220          INCW    UCB$W_RWAITCNT(R5)              ; & bumped bit accordingly.
07 12 A3   03   E0   0356  1221  125$:   BBS     #CDDB$V_RECONNECT, -            ; If reconnect in progress,
                     035B  1222                  CDDB$W_STATUS(R3), 130$         ; skip wait count re adjustment.
       56 A5   B7    035B  1223          DECW    UCB$W_RWAITCNT(R5)              ; Else, decrement bumped count
                     035E  1224          ASSUME  UCB$V_MSCP_WAITBMP GE 8         ; and clear wait count bumped
   69 A5   04   8A   035E  1225          BICB    #<UCB$M_MSCP_WAITBMP @ -8>, -   ; bit.
                     0362  1226                  UCB$W_DEVSTS+1(R5)
                     0362  1227
24 A5   18 A3   D0   0362  1228  130$:   MOVL    CDDB$L_CRB(R3), UCB$L_CRB(R5)   ; Setup new CRB address in UCB.
          03A5   30   0367  1229          BSBW    DUTU$LINK_UCB2CDDB              ; Link UCB into new CDDB chain.
                     036A  1230
                     036A  1231          ASSUME  UCB$V_MSCP_FLOVR GE 8           ; Switch UCB failed-over flag
   69 A5   08   8C   036A  1232          XORB    #<UCB$M_MSCP_FLOVR @ -8>, -     ; state.
                     036E  1233                  UCB$W_DEVSTS+1(R5)
0C 12 A3   0B   E0   036E  1234          BBS     #CDDB$V_2PBSY, -                ; Branch if I/O database update
                     0373  1235                  CDDB$W_STATUS(R3), 140$         ; fork thread is already active.
                     0373  1236          CREATE_FORK   DUTU$FAILOVER_IODB -       ; Else, make the fork thread
                     0373  1237                  frkblk = CDDB$A_2PFRB(R3)       ; active.
                     037F  1238
51   00C0 C5   3C   C1   037F  1239  140$:   ADDL3   #CDDB$L_RSTRTQFL, -          ; Get address of failed CDDB
                     0385  1240                  UCB$L_2P_CDDB(R5), R1           ; restart queue header.
                     0385  1241          ASSUME  CDRP$L_FQFL EQ 0                ; Now form "previous" entry
   50   51   D0    0385  1242          MOVL    R1, R0                          ; address for queue scan.
52   4C A5   DE    0388  1243          MOVAL   UCB$L_IOQFL(R5), R2             ; Init prev. entry for INSQUE.
   50   60   D0    038C  1244  150$:   MOVL    CDRP$L_FQFL(R0), R0             ; Link to next restart CDRP.
   51   50   D1    038F  1245          CMPL    R0, R1                          ; Is this the end?
        13   13    0392  1246          BEQL    570$                            ; Branch if this is the end.
55   BC   A0   D1    0394  1247          CMPL    CDRP$L_UCB(R0), R5              ; Is this CDRP for this UCB?
        F2   12    0398  1248          BNEQ    150$                            ; Branch if not our CDRP.
   50   60   0F    039A  1249          REMQUE  (R0), R0                        ; Else, de-queue the CDRP.
52   A0   A0   0E    039D  1250          INSQUE  CDRP$L_IOQFL(R0), (R2)          ; Add IRP to UCB I/O queue.
52   A0   A0   DE    03A1  1251          MOVAL   CDRP$L_IOQFL(R0), R2            ; Make this IRP predecessor
                     03A5  1252                                                  ; to the next.
        E5   11    03A5  1253          BRB     150$                            ; Go look for another CDRP.
                     03A7  1254
   50   01   D0    03A7  1255  570$:   MOVL    #SS$_NORMAL, R0                 ; Indicate success.
        3E   BA    03AA  1256  580$:   POPR    #^M<R1,R2,R3,R4,R5>             ; Restore saved registers.
             05    03AC  1257          RSB                                     ; Exit.
```

```
                        03AD  1259          .SBTTL  DUTU$FAILOVER_IODB - Update DDB I/O database after a failover
                        03AD  1260 ;++
                        03AD  1261 ;
                        03AD  1262 ; DUTU$FAILOVER_IODB - Update DDB I/O database after a failover
                        03AD  1263 ;
                        03AD  1264 ; Functional Description:
                        03AD  1265 ;
                        03AD  1266 ;       Since changes in the DDB - UCB chains in the I/O database must wait
                        03AD  1267 ;       until the I/O database mutex is unowned and since such changes are not
                        03AD  1268 ;       required for performing I/O requests by the class drivers, that I/O
                        03AD  1269 ;       database update is performed (and maybe postponed) in this fork
                        03AD  1270 ;       routine.  This routine uses a one-per-CDDB fork block.  Therefore, it
                        03AD  1271 ;       cannot failover just one UCB.  It must scan all UCBs chained to the
                        03AD  1272 ;       CDDB and failover all UCBs requiring such action.
                        03AD  1273 ;
                        03AD  1274 ;       After write access to the I/O database mutex is assured, all UCBs on
                        03AD  1275 ;       the input CDDB requiring DDB-failover are so altered.  The failover is
                        03AD  1276 ;       accomplished by unlinking the UCB from both the primary and the
                        03AD  1277 ;       secondary DDB chains, switching primary and secondary DDBs, and
                        03AD  1278 ;       relinking the UCB into the two (switched) chains.
                        03AD  1279 ;
                        03AD  1280 ; Inputs:
                        03AD  1281 ;
                        03AD  1282 ;       R3      CDDB address
                        03AD  1283 ;       R5      Failover (2P) fork block address
                        03AD  1284 ;
                        03AD  1285 ; Outputs:
                        03AD  1286 ;
                        03AD  1287 ;       R0 - R5 are destroyed.
                        03AD  1288 ;       All other registers are preserved.
                        03AD  1289 ;
                        03AD  1290 ; Special Notes:
                        03AD  1291 ;
                        03AD  1292 ;       This routine is coded so as not to assume that any UCB with the
                        03AD  1293 ;       UCB$V_MSCP_FLOVR flag set must have its DDB chains switched.
                        03AD  1294 ;       Currently, the two-path-per-device restriction plus the coding which
                        03AD  1295 ;       sets UCB$V_MSCP_FLOVR in DUTU$FAILOVER_UCB do guarantee this to be the
                        03AD  1296 ;       case.  However, may future plans indicate that such guarantees may not
                        03AD  1297 ;       always be possible.  Therefore, this routine is coded is a cautious
                        03AD  1298 ;       manner.
                        03AD  1299 ;--
                        03AD  1300
                        03AD  1301          ASSUME  CDDB$V_2PBSY GE 8
                        03AD  1302
                        03AD  1303 DUTU$FAILOVER_IODB:
                        03AD  1304
    13 A3   08   88     03AD  1305          BISB    #<CDDB$M_2PBSY @ -8>, - ; Set fork block busy flag.
                        03B1  1306                  CDDB$W_STATUS+1(R3)
                        03B1  1307          WAIT_FOR_IODB                   ; Obtain I/O database write access.
    55   84 A3   9E     03D1  1308          MOVAB   <CDDB$L_UCBCHAIN -      ; Initialize "previous" UCB address.
                        03D5  1309                  -UCB$L_CDDB_LINK>(R3), R5
         3D'AF   9F     03D5  1310          PUSHAB  B^999$                  ; Setup bug-trap for forking FIND_DDB.
                        03D8  1311
    55   00C4 C5   D0   03D8  1312 10$:     MOVL    UCB$L_CDDB_LINK(R5), R5 ; Link to next UCB.
              4F   13   03DD  1313          BEQL    90$                     ; Exit if no more UCBs to process.
F4 68 A5   0B   E5     03DF  1314          BBCC    #UCB$V_MSCP_FLOVR, -    ; Branch if UCB not failed-over and
                        03E4  1315                  UCB$W_DEVSTS(R5), 10$   ; clear failed-over bit.
```

```
                54   1C A3   D0  03E4  1316          MOVL    CDDB$L_DDB(R3), R4      ; Setup starting DDB for DUTU$FIND_DDB.
          53    28 A5   14   C1  03E8  1317          ADDL3   #DDB$T_NAME, -          ; Setup device name address string for
                                03ED  1318                  UCB$L_DDB(R5), R3       ; DUTU$FIND_DDB.
                      0242   30  03ED  1319          BSBW    DUTU$FIND_DDB           ; Find right DDB on this CDDB for UCB.
                 28 A5   54   D1  03F0  1320          CMPL    R4, UCB$L_DDB(R5)       ; Is correct DDB already primary?
                         E2   13  03F4  1321          BEQL    10$                     ; Branch if correct DDB already setup.
              00A0 C5    54   D1  03F6  1322          CMPL    R4, UCB$L_2P_DDB(R5)    ; Is found DDB the secondary DDB?
                         3C   12  03FB  1323          BNEQ    998$                    ; If not, something is very wrong.
               00000000'GF    16  03FD  1324          JSB     G^IOC$SEVER_UCB         ; Unlink UCB from primary DDB chain.
                      0331   30  0403  1325          BSBW    DUTU$SEVER_SEC_UCB      ; Unlink UCB from secondary DDB chain.
                                0406  1326          SWITCH  UCB$L_DDB(R5), -        ; Switch primary and secondary DDBs.
                                0406  1327                  UCB$L_2P_DDB(R5)
                   52   55   D0  0415  1328          MOVL    R5, R2                  ; Copy UCB address for relinking.
               00000000'GF    16  0418  1329          JSB     G^IOC$LINK_UCB          ; Relink to primary DDB chain.
                     14 50    E9  041E  1330          BLBC    R0, 997$                ; Branch if relinking error.
                      02BA   30  0421  1331          BSBW    DUTU$LINK_SEC_UCB       ; Relink to secondary DDB chain.
                     0E 50    E9  0424  1332          BLBC    R0, 997$                ; Branch if relinking error.
          53    00BC C5    D0  0427  1333          MOVL    UCB$L_CDDB(R5), R3      ; Guarantee a correct CDDB address.
                         AA   11  042C  1334          BRB     10$                     ; Loop till no more UCBs to process.
                                042E  1335
                         8E   D5  042E  1336  90$:   TSTL    (SP)+                   ; Remove DUTU$FIND_DDB bug-trap.
                 13 A3   08   8A  0430  1337          BICB    #<CDDB$M_2PBSY @ -8>, - ; Clear fork block busy flag.
                                0434  1338                  CDDB$W_STATUS+1(R3)
                         05   0434  1339          RSB                             ; End fork thread.
                                0435  1340
                                0435  1341  ; A duplicate unit number has been discovered during the primary or secondary
                                0435  1342  ; DDB relinking attempt.
                                0435  1343
                                0435  1344  997$:   BUG_CHECK INCONSTATE, FATAL
                                0439  1345
                                0439  1346  ; The DDB found by DUTU$FIND_DDB is neither the primary nor the secondary DDB
                                0439  1347  ; for this UCB.  In this case, something is totally confused because there
                                0439  1348  ; should only be two CDDBs per UCB.  If more than two paths are allow for a
                                0439  1349  ; single device, some other action would be required at this point.
                                0439  1350
                                0439  1351  998$:   BUG_CHECK MSCPCLASS, FATAL
                                043D  1352
                                043D  1353  ; DUTU$FIND_DDB has forked.  This should never happen.  FIND_DDB forks because
                                043D  1354  ; the I/O database in not available for write access or because it cannot
                                043D  1355  ; allocate pool for an new DDB.  However, this routine already has write
                                043D  1356  ; access to the I/O database mutex and the DDB which FIND_DDB is "looking" for
                                043D  1357  ; should already exist.  Therefore, DUTU$FIND_DDB forking is a very fatal
                                043D  1358  ; error.  Also note: that the MOVL used to restore the CDDB address just
                                043D  1359  ; before the BRB 10$ would be invalid should the DUTU$FIND_DDB actually fork.
                                043D  1360
                                043D  1361  999$:   BUG_CHECK MSCPCLASS, FATAL
```

```
0441  1363          .SBTTL  ----- NEW UNIT ROUTINES -----
0441  1364          .SBTTL  DUTU$NEW_UNIT - Process a possible new unit
0441  1365 ;++
0441  1366 ;
0441  1367 ; DUTU$NEW_UNIT - Process a possible new unit
0441  1368 ;
0441  1369 ; Functional Description:
0441  1370 ;
0441  1371 ;       This routine processes a unit available attention or get unit status
0441  1372 ;       end message.  If the I/O database does not reflect the presence of the
0441  1373 ;       unit described in the MSCP message, the new unit is added to the I/O
0441  1374 ;       database in whatever manner is appropriate.  The only condition which
0441  1375 ;       will prohibit adding the new unit is insufficient non-paged pool to
0441  1376 ;       accomidate the UCB.  When this occurs, the message is ignored.
0441  1377 ;
0441  1378 ;       First a check is made to determine whether the unit is already known
0441  1379 ;       to the I/O database.  The unit can be either on a primary or a
0441  1380 ;       secondary path.  If the unit belongs on a secondary path, the routine
0441  1381 ;       which checks for a previous secondary path condition will
0441  1382 ;       automatically place the unit on the secondary path.
0441  1383 ;
0441  1384 ;       If no primary or secondary UCB can be located, a new UCB is allocated.
0441  1385 ;       If this allocation fails, control is immediately returned to the
0441  1386 ;       caller and no further attempt is made to represent the new unit in the
0441  1387 ;       I/O database.
0441  1388 ;
0441  1389 ;       Upon successful allocation and setup by IOC$COPY_UCB, this routine
0441  1390 ;       fills in a few more fields in the new UCB.  The most important fields
0441  1391 ;       are those which are copies of the information in the MSCP message.
0441  1392 ;       This is the last opportunity in the UCB creation process to reference
0441  1393 ;       the MSCP message.  Once all the useful information has been copied
0441  1394 ;       from the MSCP message to the new UCB, a fork thread is started to
0441  1395 ;       complete initialization of the UCB and link it into the I/O database.
0441  1396 ;
0441  1397 ;       Whether or not it has been completely setup, the new UCB address is
0441  1398 ;       returned in R2.
0441  1399 ;
0441  1400 ; Inputs:
0441  1401 ;
0441  1402 ;       R1      size of the MSCP message
0441  1403 ;       R2      base address of the MSCP message
0441  1404 ;       R3      CDDB address
0441  1405 ;
0441  1406 ;
0441  1407 ; Implicit Inputs:
0441  1408 ;
0441  1409 ;       Template UCB            in .PSECT $$$200_TEMPLATE_UCB_01
0441  1410 ;       Template ORB            in .PSECT $$$200_TEMPLATE_ORB_01
0441  1411 ;       DUTU$L_CDDB_LISTHEAD    location containing the system virtual address
0441  1412 ;                               of the CDDB listhead for this device class
0441  1413 ;
0441  1414 ; Outputs:
0441  1415 ;
0441  1416 ;       R2      UCB address for unit represented in the MSCP message (either
0441  1417 ;               new or already existing)
0441  1418 ;
0441  1419 ;       R0 through R2 destroyed.
```

```
                     ;       All other registers preserved.
                     ;
                     ; WARNINGS:
                     ;
                     ;       The UCB address returned by this routine may or may not be correctly
                     ;       linked into the I/O database.  The linking operation is performed in a
                     ;       seperate fork thread (which uses the UCB as a fork block).  This fork
                     ;       thread can become stalled due to lack of memory needed for creation of
                     ;       a DDB, due to lack of write access to the I/O database, or possibly
                     ;       some other reason.  The only thing which is certain is that at some
                     ;       time (now or in the future) the new unit will be represented by the
                     ;       UCB whose address is returned in R2.
                     ;
                     ; FORKING CONSIDERATIONS:
                     ;
                     ;       The UCB is currently used as a fork block by the class drivers in
                     ;       three different instances:
                     ;
                     ;           1.  Creation of the CDDB when the controller initialization
                     ;               routine is called.
                     ;           2.  For the fork thread which links a new UCB into the I/O
                     ;               database.
                     ;           3.  For the fork thread which links the secondary path threads to
                     ;               a UCB which has been determined to represent a dual-pathed
                     ;               device.
                     ;
                     ;       Athough it is not obvious, these three uses cannot be competing to use
                     ;       the same UCB as a fork block concurrently.  Usage 1 occurs only when
                     ;       no connection to the MSCP server exists.  At that time, no MSCP
                     ;       messages can be received.  Therefore, usage 1 precludes usages 2 and
                     ;       3.  While the fork block is in type 2 usage, the UCB is unknown to the
                     ;       I/O database.  Therefore, it cannot be detected and put into use by a
                     ;       type 3 fork thread.  Thus usages 2 and 3 are mutually exclusive.
                     ;
                     ;       There is a race condition (which the LINK_NEW_UCB fork thread must
                     ;       detect).  This race results from a stalled new UCB thread preventing
                     ;       detection of a dual path condition.  Before actually linking a new UCB
                     ;       into the I/O database but after all possible stall conditions have
                     ;       been delt with, the new UCB fork thread must make one last attempt to
                     ;       dualpath link the new UCB.
                     ;--
                     ;
                     ; Locate the template UCB and ORB
                     ;
                     
                             .SAVE
                             .PSECT  $$$200_TEMPLATE_UCB_00 RD,WRT,EXE,LONG
                     DUTU$TEMPLATE_UCB::
                     
                             .PSECT  $$$200_TEMPLATE_ORB_00 RD,WRT,EXE,LONG
                     DUTU$TEMPLATE_ORB::
                     
                             .RESTORE
                     
                     DUTU$NEW_UNIT::
```

```
                        0441  1477
04 A2   0FFF 8F   B1    0441  1478          CMPW    #4095, MSCP$W_UNIT(R2)  ; KLUDGE ALERT:  Is the unit to be
                  13    0447  1479          BEQL    19$                     ; created 4095?  If so, don't.
                010F    0449  1480          BSBW    DUTU$LOOKUP_UCB         ; Do we already know of this unit?
                  07    044C  1481          BNEQ    15$                     ; Branch if unit already known.
                012A    044E  1482          BSBW    DUTU$SETUP_DUAL_PATH    ; Is this unit, or should this unit
                        0451  1483                                          ; be, dual pathed?
                  50    D5    0451  1484          TSTL    R0                ; Was unit dual pathed?
                  0A    13    0453  1485          BEQL    30$               ; Branch if unit not dual pathed.
              52  50    D0    0455  1486  15$:     MOVL    R0, R2            ; Copy address of previous UCB.
                        05    0458  1487          RSB                       ; Return to caller with address of
                        0459  1488                                          ; previously existant UCB in R2.
                        0459  1489
              54  8E    7D    0459  1490  18$:     MOVQ    (SP)+, R4         ; Restore saved registers.
                  52    D4    045C  1491  19$:     CLRL    R2                ; Indicate new UCB cannot be created.
                        05    045E  1492          RSB                       ; Exit without creating new UCB.
                        045F  1493
                0A4B    30    045F  1494  30$:     BSBW    DUTU$FILL_MSCP_MSG  ; Zero extend the MSCP message.
              7E  54    7D    0462  1495          MOVQ    R4, -(SP)         ; Save a couple of registers.
              54  52    D0    0465  1496          MOVL    R2, R4            ; Copy MSCP message address.
    55  0000'CF    9E    0468  1497          MOVAB   W^DUTU$TEMPLATE_UCB, R5 ; Get template UCB address.
1C A5   0000'CF    9E    046D  1498          MOVAB   W^DUTU$TEMPLATE_ORB, -  ; Set template ORB address.
                        0473  1499                  UCB$L_ORB(R5)
    00000000'GF     16  0473  1500          JSB     G^IOC$COPY_UCB          ; Make new UCB from template.
                DD  50  E9    0479  1501          BLBC    R0, 18$           ; Branch if new UCB not created.
                        047C  1502
                        047C  1503                                          ; Compensate for unwanted IOC$COPY_UCB
                        047C  1504                                          ; initialization:
  56 A2   56 A5     B0    047C  1505          MOVW    UCB$W_RWAITCNT(R5), -   ; - Restore RWAITCNT.
                        0481  1506                  UCB$W_RWAITCNT(R2)
  5C A2   5C A5     B0    0481  1507          MOVW    UCB$W_REFC(R5), -       ; - Restore reference count.
                        0486  1508                  UCB$W_REFC(R2)
  68 A2   68 A5     B0    0486  1509          MOVW    UCB$W_DEVSTS(R5), -     ; - Restore device dependent status.
                        048B  1510                  UCB$W_DEVSTS(R2)
                        048B  1511
                        048B  1512          ASSUME  UCB$L_MAXBLOCK EQ -     ; Since no I/O will go to this "disk"
                        048B  1513                  UCB$L_RECORD            ; before the first PACKACK, zero the
          00B0  C2  D4  048B  1514          CLRL    UCB$L_MAXBLOCK(R2)      ; maximum block number (for tapes this
                        048F  1515                                          ; harmlessly zeros the frame number).
                        048F  1516
                        048F  1517                                          ; Initialize fields required by the
                        048F  1518                                          ; final UCB setup fork thread:
008C C2   1C A4     D0  048F  1519          MOVL    MSCP$L_MEDIA_ID(R4), -  ;  - MSCP media identification.
                        0495  1520                  UCB$L_MEDIA_ID(R2)
00D4 C2   04 A4     B0  0495  1521          MOVW    MSCP$W_UNIT(R4), -      ;  - MSCP unit number.
                        049B  1522                  UCB$W_MSCPUNIT(R2)
                        049B  1523
                        049B  1524          CREATE_FORK -                   ; Start a fork thread to complete
                        049B  1525                  LINK_NEW_UCB, -         ; setup and linking of the new UCB
                        049B  1526                  frkb[k = (R2), -        ; into the I/O database.
                        049B  1527                  mask = <^M<R2,R3>>
                        04A5  1528
              54  8E    7D    04A5  1529          MOVQ    (SP)+, R4         ; Restore saved R4 and R5.
                        05    04A8  1530          RSB                       ; Return to caller with new UCB
                        04A9  1531                                          ; address in R2.
```

```
;++
;
; LINK_NEW_UCB
;
; Functional Description:
;
;       Acting in the context of an independent fork thread and using the UCB
;       as a fork block this routine completes initialization of a new UCB and
;       establishes the primary path linkage for that UCB.
;
;       The various backpointer fields in the UCB are filled in.  Previously
;       uninitialized queue headers are initialized.  Where appropriate, state
;       bits are altered.  If possible, a device type is determined.  The name
;       of the device (DDCn form) is established.
;
;       A suitable DDB is located or created.  Then write access to the I/O
;       database is obtained.  Either one or both of these operations may
;       cause the thread to fork.
;
;       After completing those operations which can fork, a final check for
;       a dual path device is made.  This eliminates a possible race between
;       two new UCB creations for what is truely a dual pathed device.  If a
;       dual path condition is found, the already existing I/O database is
;       altered to reflect the dual path device and the UCB used as a fork
;       block for this thread (and the thread itself) are discarded.
;
;       If the dual path lookup fails, the new UCB is linked into the I/O and
;       class driver databases and some connection dependent fields are
;       initialized to reflect the current connection to the actual device.
;
; Inputs:
;
;       R3      CDDB address
;       R5      UCB address (also used as a fork block)
;       fork context at IPL$_SCS
;
; Implicit Inputs:
;
;       UCB$W_MSCPUNIT(R5)      MSCP unit number for device
;       UCB$L_MEDIA_ID(R5)      MSCP media identification for device
;
; Outputs: None.
;
; Implicit Outputs:
;
;    Backpointers:
;       UCB$L_CRB(R5)           CRB address
;       UCB$L_DDT(R5)           DDT address
;
;    I/O Database and Class Driver Linkages:
;       UCB$W_UNIT(R5)          I/O database unit number
;       UCB$L_DDB(R5)           DDB address
;       UCB$L_LINK(R5)          Link to next UCB in DDB chain
;       UCB$L_CDDB(R5)          CDDB address
;       UCB$L_CDDB_LINK(R5)     Link to next UCB in CDDB chain
;
;    Connection Dependent:
```

```
                     04A9  1590 ;       UCB$L_CDT(R5)           CDT address
                     04A9  1591 ;       UCB$L_PDT(R5)           PDT address
                     04A9  1592 ;
                     04A9  1593 ;    Other:
                     04A9  1594 ;       UCB$W_DEVSTS(R5)        UCB$M_MSCP_INITING cleared
                     04A9  1595 ;       UCB$L_MAXBCNT(R5)       copied from PDT$L_MAXBCNT(pdt)
                     04A9  1596 ;
                     04A9  1597 ;       if CDDB is not initializing or reconnecting:
                     04A9  1598 ;         UCB$W_DEVSTS(R5)      UCB$M_MSCP_WAITBMP cleared
                     04A9  1599 ;         UCB$W_RWAITCNT(R5)    decremented
                     04A9  1600 ;
                     04A9  1601 ; WARNINGS:
                     04A9  1602 ;
                     04A9  1603 ;       The CDDB$L_DDB in this CDDB must point to at least one valid DDB for
                     04A9  1604 ;       the devices which are (could be) accessed via that CDDB.  The major
                     04A9  1605 ;       implication here is that we can never discard the DU or TU DDB handed
                     04A9  1606 ;       us by SYSGEN, et. al. when calling the driver at its controller
                     04A9  1607 ;       initialization entry point.
                     04A9  1608 ;--
                     04A9  1609
                     04A9  1610 LINK_NEW_UCB:
                     04A9  1611
                     04A9  1612         ; Guarantee sufficient space for device name formation.
                     04A9  1613         ASSUME  DDB$S_NAME EQ 16
                     04A9  1614         .IF     DEFINED UCB$L_CANLINK
                     04A9  1615         ASSUME  UCB$L_CDT     EQ <UCB$L_CDDB_LINK+4>
                     04A9  1616         ASSUME  UCB$L_CANLINK EQ <UCB$L_CDDB_LINK+8>
                     04A9  1617         ASSUME  UCB$Q_UNIT_ID EQ <UCB$L_CDDB_LINK+12>
                     04A9  1618         .IF FALSE
                     04A9  1619         ASSUME  UCB$L_CDT     EQ <UCB$L_CDDB_LINK+4>
                     04A9  1620         ASSUME  UCB$S_UNIT_ID EQ 8
                     04A9  1621         ASSUME  UCB$Q_UNIT_ID EQ <UCB$L_CDDB_LINK+8>
                     04A9  1622         .ENDC
00BC C5    53    D0  04A9  1623         MOVL    R3, UCB$L_CDDB(R5)              ; Save CDDB address.
  54    1C A3    D0  04AE  1624         MOVL    CDDB$L_DDB(R3), R4              ; Get beginning DDB for scan.
53    00C4 C5    9E  04B2  1625         MOVAB   UCB$L_CDDB_LINK(R5), R3         ; Begin device name setup.
        02BB      30  04B7  1626         BSBW    DUTU$GET_DEVNAM                 ; Determine device name & unit.
        0175      30  04BA  1627         BSBW    DUTU$FIND_DDB                   ; Lookup that device name.
  28 A5    54    D0  04BD  1628         MOVL    R4, UCB$L_DDB(R5)               ; Save DDB address.
     00C4 C5    7C  04C1  1629         CLRQ    UCB$L_CDDB_LINK(R5)             ; Cleanup after name builder.
     00CC C5    7C  04C5  1630         CLRQ    UCB$L_CDDB_LINK+8(R5)
                     04C9  1631
                     04C9  1632         WAIT_FOR_IODB                           ; Obtain write access to the
                     04E9  1633                                                 ; I/O database.
                     04E9  1634
                     04E9  1635         ; The following hack allows DUTU$SETUP_DUAL_PATH to be used for this
                     04E9  1636         ; last chance dual path lookup.  It depends DUTU$LOOKUP_UCB using only
                     04E9  1637         ; the MSCP$W_UNIT field of a MSCP message.
     51    06    D0  04E9  1638         MOVL    #MSCP$W_UNIT+2, R1              ; Fake MSCP message size.
52   00D0 C5    9E  04EC  1639         MOVAB   <UCB$W_MSCPUNIT -               ; Fake MSCP message address.
                     04F1  1640                 -MSCP$W_UNIT>(R5), R2
53   00BC C5    D0  04F1  1641         MOVL    UCB$L_CDDB(R5), R3              ; Get CDDB address too.
        0082      30  04F6  1642         BSBW    DUTU$SETUP_DUAL_PATH            ; Make last chance check for
          50      D5  04F9  1643         TSTL    R0                              ; a dual pathed device.
          54      12  04FB  1644         BNEQ    700$                            ; Branch if dual path found.
                     04FD  1645
     53    55    D0  04FD  1646         MOVL    R5, R3                          ; Copy UCB address.
```

```
                02F5   30  0500  1647          BSBW    DUTU$GET_DEVTYPE                ; Determine device type.
          53  00BC C5   D0  0503  1648          MOVL    UCB$L_CDDB(R5), R3              ; Restore CDDB address.
    0088 C5    0C A4   D0  0508  1649          MOVL    DDB$L_DDT(R4), UCB$L_DDT(R5)    ; Setup DDT address.
       24 A5    18 A3   D0  050E  1650          MOVL    CDDB$L_CRB(R3), UCB$L_CRB(R5)   ; Setup CRB address.
                        0513  1651
                030F   30  0513  1652          BSBW    DUTU$INIT_CONN_UCB              ; Initialize connection
                        0516  1653                                                      ; dependent fields.
          50  0084 C5   D0  0516  1654          MOVL    UCB$L_PDT(R5), R0               ; Get PDT address.
    00B4 C5   00BC C0   D0  051B  1655          MOVL    PDT$L_MAXBCNT(R0), -            ; Copy maximum bytes per
                        0522  1656                  UCB$L_MAXBCNT(R5)                   ; count to UCB.
                        0522  1657
                        0522  1658          ; N.B. all actions which alter the condition of this UCB with respect
                        0522  1659          ; to the remainder of the I/O database MUST occur following the call
                        0522  1660          ; to IOC$LINK_UCB below.  This results from the possibility that
                        0522  1661          ; IOC$LINK_UCB will determine that this UCB is a duplicate, afterwhich
                        0522  1662          ; this UCB will be deallocated.
                        0522  1663
             52   55    D0  0522  1664          MOVL    R5, R2                          ; Copy UCB address.
     00000000'GF   16    0525  1665          JSB     G^IOC$LINK_UCB                  ; Link UCB to primary DDB.
           23 50   E9    052B  1666          BLBC    R0, 700$                        ; If UCB is duplicate, quit.
                01DE   30  052E  1667          BSBW    DUTU$LINK_UCB2CDDB              ; Link UCB into CDDB chain.
                        0531  1668
          53  00BC C5   D0  0531  1669          MOVL    UCB$L_CDDB(R5), R3              ; Restore CDDB address.
                02FA   30  0536  1670          BSBW    DUTU$SETUP_CDP_UCB              ; If needed, setup local
                        0539  1671                                                      ; UCB to class driver UCB
                        0539  1672                                                      ; dual path.
                        0539  1673
       12 A3    0C   B3  0539  1674          BITW    #<CDDB$M_INITING -              ; Is init or reconnect "still"
                        053D  1675                    !CDDB$M_RECONNECT>, -           ; in progress?
                        053D  1676                  CDDB$W_STATUS(R3)                 ; If so somebody else will fix
                    0B  12  053D  1677          BNEQ    90$                             ; RWAITCNT and we can skip it.
    68 A5    0400 8F   AA  053F  1678          BICW    #UCB$M_MSCP_WAITBMP, -          ; Else, indicate RWAITCNT no
                        0545  1679                  UCB$W_DEVSTS(R5)                  ; longer bumped.
             56 A5   B7  0545  1680          DECW    UCB$W_RWAITCNT(R5)              ; Decrement wait count.
                    0D  12  0548  1681          BNEQ    888$                            ; It had better be zero.
                        054A  1682
    68 A5    0200 8F   AA  054A  1683  90$:   BICW    #UCB$M_MSCP_INITING, -          ; Clear the "initing" flag.
                        0550  1684                  UCB$W_DEVSTS(R5)
                        0550  1685
                    05  0550  1686          RSB                                     ; dade, dade, dade; that's all
                        0551  1687                                                  ; folks.
                        0551  1688
          50   55   D0  0551  1689  700$:  MOVL    R5, R0                          ; Found dual path or dup. UCB:
               FBF8 31  0554  1690          BRW     DEANONPAGED                     ; deallocate this UCB and
                        0557  1691                                                  ; discontinue fork thread.
                        0557  1692
                        0557  1693  888$:  BUG_CHECK MSCPCLASS, FATAL               ; RWAITCNT not zero at end of
                        055B  1694                                                  ; new UCB setup.
```

```
                     055B  1696          .SBTTL  DUTU$LOOKUP_UCB - Locate a given MSCP unit on given CDDB chain
                     055B  1697 ;++
                     055B  1698 ;
                     055B  1699 ; DUTU$LOOKUP_UCB - Locate a given MSCP unit on given CDDB chain
                     055B  1700 ;
                     055B  1701 ; Functional Description:
                     055B  1702 ;
                     055B  1703 ;       The chain of UCBs linked to the CDDB whose address is input to this
                     055B  1704 ;       routine is scanned for a UCB containing a MSCP unit number matching
                     055B  1705 ;       the one in the input MSCP message.  If such a match is found, the
                     055B  1706 ;       address of the UCB is returned in R0.  Otherwise, R0 is returned as
                     055B  1707 ;       zero.
                     055B  1708 ;
                     055B  1709 ; Inputs:
                     055B  1710 ;
                     055B  1711 ;       R1      size of the MSCP message
                     055B  1712 ;       R2      base address of the MSCP message
                     055B  1713 ;       R3      CDDB address
                     055B  1714 ;
                     055B  1715 ; Outputs:
                     055B  1716 ;
                     055B  1717 ;       R0      address of the UCB with matching MSCP unit number, or zero
                     055B  1718 ;
                     055B  1719 ;       condition-code
                     055B  1720 ;         EQL ==> no match found [R0 EQL 0]
                     055B  1721 ;         NEQ ==> match found [R0 NEQ 0]
                     055B  1722 ;
                     055B  1723 ;       All registers except R0 are preserved
                     055B  1724 ;--
                     055B  1725
                     055B  1726 DUTU$LOOKUP_UCB::
                     055B  1727
          06   51    B1  055B  1728          CMPW    R1, #MSCP$W_UNIT+2              ; Is message big enough to
               18    1F  055E  1729          BLSSU   90$                             ; contain a unit; branch if not.
                         0560  1730
       50   84  A3    9E  0560  1731          MOVAB   <CDDB$L_UCBCHAIN -              ; Initialize previous UCB
                         0564  1732                  -UCB$L_CDDB_LINK>(R3), R0       ; address.
                         0564  1733
       50  00C4 C0    D0  0564  1734 10$:     MOVL    UCB$L_CDDB_LINK(R0), R0         ; Link to next UCB.
                   0D    13  0569  1735          BEQL    90$                             ; Branch if no more UCBs.
 04 A2 00D4 C0    B1  056B  1736          CMPW    UCB$W_MSCPUNIT(R0), -           ; Is the MSCP unit number
                         0571  1737                  MSCP$W_UNIT(R2)                 ; right?
                   F1    1F  0571  1738          BLSSU   10$                             ; If wrong, loop if still worth
                   03    1A  0573  1739          BGTRU   90$                             ; looking; else, exit w/ error.
                         0575  1740
                   50    D5  0575  1741          TSTL    R0                              ; If good, set condition code
                   05    0577  1742          RSB                                     ; and return.
                         0578  1743
                   50    D4  0578  1744 90$:     CLRL    R0                              ; Signal UCB not found.
                   05    057A  1745          RSB                                     ; Return.
```

```
                        057B  1747          .SBTTL  DUTU$SETUP_DUAL_PATH - Reflect dual path access in I/O database
                        057B  1748 ;++
                        057B  1749 ;
                        057B  1750 ; DUTU$SETUP_DUAL_PATH - Reflect dual path access in I/O database
                        057B  1751 ;
                        057B  1752 ; Functional Description:
                        057B  1753 ;
                        057B  1754 ;       All UCBs linked to all CDDBs chained to the listhead given in R0
                        057B  1755 ;       (except the CDDB whose address is given in R3) is scanned for a UCB
                        057B  1756 ;       whose allocation class matches that of the CDDB pointed to by R3 and
                        057B  1757 ;       whose MSCP unit number matches the one in the input MSCP message.  If
                        057B  1758 ;       such UCB is found and has not already been dual path chained, the UCB
                        057B  1759 ;       will be dual path chained to input CDDB and its related I/O database.
                        057B  1760 ;       If the UCB is already chained to the input CDDB, no action is taken.
                        057B  1761 ;       If the UCB is already chained to some other CDDB, an "Inconsistant I/O
                        057B  1762 ;       database" bugcheck is generated.  If a UCB is found, its address is
                        057B  1763 ;       returned in R0.  Otherwise, R0 is returned as zero.
                        057B  1764 ;
                        057B  1765 ; Inputs:
                        057B  1766 ;
                        057B  1767 ;       R1      size of the MSCP message
                        057B  1768 ;       R2      base address of the MSCP message
                        057B  1769 ;       R3      CDDB address
                        057B  1770 ;
                        057B  1771 ; Implicit Inputs:
                        057B  1772 ;
                        057B  1773 ;       DUTU$L_CDDB_LISTHEAD    location containing the system virtual address
                        057B  1774 ;                               of the CDDB listhead for this device class
                        057B  1775 ; Outputs:
                        057B  1776 ;
                        057B  1777 ;       R0      address of the dual path UCB, or zero
                        057B  1778 ;
                        057B  1779 ;       All registers except R0 are preserved
                        057B  1780 ;
                        057B  1781 ; WARNINGS:
                        057B  1782 ;
                        057B  1783 ;       This routine does not scan the local node I/O database.  Therefore,
                        057B  1784 ;       only dual path devices which have both paths served by the disk or
                        057B  1785 ;       tape class driver will be found by this routine.  This feature is not
                        057B  1786 ;       needed until the MSCP server becomes capable of issuing access path
                        057B  1787 ;       attention messages.
                        057B  1788 ;
                        057B  1789 ;       The dual path nature of the device may not be reflected in the I/O
                        057B  1790 ;       database immediately upon exit from this routine.  That operation is
                        057B  1791 ;       performed in a fork thread which can become stalled due to lack of
                        057B  1792 ;       non-paged pool or inablilty to obtain write access to the I/O
                        057B  1793 ;       database.  When this routine exits, however, the fork thread has been
                        057B  1794 ;       established and it will complete at some future time.
                        057B  1795 ;
                        057B  1796 ;--
                        057B  1797
                        057B  1798 DUTU$SETUP_DUAL_PATH::
                        057B  1799
7E  53  7D              057B  1800          MOVQ    R3, -(SP)                   ; Save a few registers.
06  51  B1              057E  1801          CMPW    R1, #MSCP$W_UNIT+2          ; Is message big enough to
    42  1F              0581  1802          BLSSU   900$                        ; contain a unit; branch if not.
                        0583  1803
```

```
                     54    53   D0   0583  1804            MOVL    R3, R4                          ; Copy input CDDB address.
          0000'CF    00000058 8F   C3   0586  1805            SUBL3   #CDDB$L_CDDBLINK, -             ; Initialize previous CDDB addr.
                              53         058F  1806                    W^<DUTU$DATA + DUTU$L_CDDB_LISTHEAD>, -
                                         0590  1807                    R3
                                         0590  1808
                     53    58   A3   D0   0590  1809 10$:     MOVL    CDDB$L_CDDBLINK(R3), R3         ; Link to next CDDB.
                                   2F   13   0594  1810            BEQL    900$                            ; Branch if no more CDDBs.
                        54         53   D1   0596  1811            CMPL    R3, R4                          ; Is it the input CDDB?
                                   F5   13   0599  1812            BEQL    10$                             ; Branch if input CDDB.
                     50    50   A3   D0   059B  1813            MOVL    CDDB$L_ALLOCLS(R3), R0          ; Can this CDDB dual path?
                                   EF   13   059F  1814            BEQL    10$                             ; Branch if it can't dual path.
                     50 A4      50   D1   05A1  1815            CMPL    R0, CDDB$L_ALLOCLS(R4)          ; Is it the right alloc. class?
                                   E9   12   05A5  1816            BNEQ    10$                             ; Branch if wrong alloc. class.
                                   B2   10   05A7  1817            BSBB    DUTU$LOOKUP_UCB                 ; Check for right UCB.
                                   E5   13   05A9  1818            BEQL    10$                             ; Branch if none found.
                                        05AB  1819
                  53    00C0 C0   D0   05AB  1820            MOVL    UCB$L_2P_CDDB(R0), R3           ; Get secondary CDDB from UCB.
                                   17   12   05B0  1821            BNEQ    930$                            ; Branch if UCB already
                                        05B2  1822                                                    ; dual pathed.
                  00C0 C0    54   D0   05B2  1823            MOVL    R4, UCB$L_2P_CDDB(R0)           ; Else, setup input CDDB as
                                        05B7  1824            CREATE_FORK -                           ; the dual path CDDB and start
                                        05B7  1825                    LINK_2P_UCB, -                  ; fork thread to dual path
                                        05B7  1826                    frkblk = (R0), -                ; link the UCB into the I/O
                                        05B7  1827                    mask = <^M<R0,R1,R2,R5>>        ; database.
                        53    8E   7D   05C1  1828 75$:     MOVQ    (SP)+, R3                       ; Restore saved registers.
                                   05   05C4  1829            RSB                                     ; Return to caller.
                                        05C5  1830
                              50   D4   05C5  1831 900$:     CLRL    R0                              ; Signal no UCB found.
                                   F8   11   05C7  1832            BRB     75$                             ; Then exit.
                                        05C9  1833
                  54    53   D1   05C9  1834 930$:     CMPL    R3, R4                          ; Is previous dual path CDDB
                                   F3   13   05CC  1835            BEQL    75$                             ; input CDDB?  Branch if yes.
                                        05CE  1836
                                        05CE  1837            BUG_CHECK IVDSKCONFG, FATAL             ; Else, bug check.
```

```
                     1839 ;++
                     1840 ;
                     1841 ; LINK_2P_UCB - Fork thread to dual path link a UCB
                     1842 ;
                     1843 ; Functional Description:
                     1844 ;
                     1845 ;       Acting in the context of an independent fork thread and using the UCB
                     1846 ;       as a fork block this routine establishes the dual path links for a
                     1847 ;       UCB.  A suitable DDB is located or created.  Write access is obtained
                     1848 ;       for the I/O database.  Finally, the UCB is dual path linked to the
                     1849 ;       suitable DDB.
                     1850 ;
                     1851 ;       This routine potentially modifies UCB$L_MAXBCNT, the field giving the
                     1852 ;       maximum number of bytes allowed in a single transfer.  The current
                     1853 ;       value in that field -- which was established when the primary path was
                     1854 ;       discovered -- is minimized with the PDT$L_MAXBCNT field in the PDT for
                     1855 ;       the secondary path.  The effect is to generally use the minimum value
                     1856 ;       accepted among the two paths on which the device is known.  A
                     1857 ;       potential problem exists if the secondary path MAXBCNT value is
                     1858 ;       smaller than the primary path value.  Some of the in progress
                     1859 ;       transfers may exceed the secondary path MAXBCNT.  Should control be
                     1860 ;       failed over to the secondary path before these transfers complete,
                     1861 ;       those transfers could not be performed.  However, this risk is deemed
                     1862 ;       minimal and will be addressed by a nonfatal bugcheck in the
                     1863 ;       appropriate port driver (at the most).
                     1864 ;
                     1865 ; Inputs:
                     1866 ;
                     1867 ;       R5      UCB address (also used as a fork block)
                     1868 ;       fork context at IPL$_SCS
                     1869 ;
                     1870 ; Implicit Inputs:
                     1871 ;
                     1872 ;       UCB$L_2P_CDDB(R5) address of the CDDB for the secondary CDDB.
                     1873 ;
                     1874 ; Outputs: None.
                     1875 ;
                     1876 ; Implicit Outputs:
                     1877 ;
                     1878 ;       UCB$L_2P_DDB(R5)  altered to point to the secondary path DDB
                     1879 ;       UCB$L_2P_LINK(R5) altered to link the UCB into the secondary chain of
                     1880 ;                         UCBs for the DDB
                     1881 ;       DEV$M_2P set in UCB$L_DEVCHAR2(R5)
                     1882 ;       UCB$L_MAXBCNT(R5)       minimized with PDT$L_MAXBCNT(pdt)
                     1883 ;
                     1884 ; WARNINGS:
                     1885 ;
                     1886 ;       The CDDB$L_DDB in this CDDB must point to at least one valid DDB for
                     1887 ;       the devices which are (could be) accessed via that CDDB.  The major
                     1888 ;       implication here is that we can never discard the DU or TU DDB handed
                     1889 ;       us by SYSGEN, et. al. when calling the driver at its controller
                     1890 ;       initialization entry point.
                     1891 ;--
                     1892
                     1893 LINK_2P_UCB:
                     1894
53   28 A5   14  C1  1895         ADDL3   #DDB$T_NAME, UCB$L_DDB(R5), R3  ; Get address of DDC name.
```

```
                50    00C0 C5    D0  05D7  1896          MOVL    UCB$L_2P_CDDB(R5), R0           ; Get sec. CDDB address.
                  54    1C A0    D0  05DC  1897          MOVL    CDDB$L_DDB(R0), R4              ; Get beginning DDB address.
                        50       10  05E0  1898          BSBB    DUTU$FIND_DDB                   ; Locate or make secondary DDB.
                00A0 C5   54     D0  05E2  1899          MOVL    R4, UCB$L_2P_DDB(R5)            ; Save secnodary DDB address.
                                     05E7  1900
                                     05E7  1901          WAIT_FOR_IODB                           ; Obtain write access to the
                                     0607  1902                                                  ; I/O database.
                                     0607  1903
                50    00C0 C5    D0  0607  1904          MOVL    UCB$L_2P_CDDB(R5), R0           ; Get secondary CDDB address.
                  50    14 A0    D0  060C  1905          MOVL    CDDB$L_PDT(R0), R0              ; Get secondary PDT address.
            00B4 C5   00BC C0    D1  0610  1906          CMPL    PDT$L_MAXBCNT(R0), -            ; Is secondary PDT bytes per
                                     0617  1907                  UCB$L_MAXBCNT(R5)               ; xfer smaller than UCB value?
                        07       1E  0617  1908          BGEQU   20$                             ; Branch if secondary larger.
            00B4 C5   00BC C0    D0  0619  1909          MOVL    PDT$L_MAXBCNT(R0), -            ; If smaller, make secondary
                                     0620  1910                  UCB$L_MAXBCNT(R5)               ; bytes per xfer the UCB value.
                                     0620  1911
                        52   55    D0  0620  1912  20$:   MOVL    R5, R2                          ; Copy UCB address for linking.
                           00B8     30  0623  1913          BSBW    DUTU$LINK_SEC_UCB               ; Make secondary DDB links.
                         05 50      E9  0626  1914          BLBC    R0, 9999$                       ; Branch if error while linking.
                    3C A5   10      C8  0629  1915          BISL    #DEV$M_2P, UCB$L_DEVCHAR2(R5)   ; Flag device as dual pathed.
                                    05  062D  1916          RSB                                     ; End fork thread.
                                     062E  1917
                                     062E  1918  9999$:  BUG_CHECK INCONSTATE, FATAL              ; Signal inconsistant I/O db.
```

```
                     0632  1920                .SBTTL  DUTU$FIND_DDB - Find or create a DDB with the right DEVNAM
                     0632  1921 ;++
                     0632  1922 ;
                     0632  1923 ; DUTU$FIND_DDB - Find or create a DDB with the right DEVNAM
                     0632  1924 ;
                     0632  1925 ; Functional Description:
                     0632  1926 ;
                     0632  1927 ;       Starting with a given DDB, this routine searches a chain of DDBs
                     0632  1928 ;       attempting to find a requested DEVNAM (DDC - device DD, controller C).
                     0632  1929 ;       If the requested DEVNAM is found, the corresponding DDB address is
                     0632  1930 ;       returned.  If the requested DEVNAM is not found, a DDB is created with
                     0632  1931 ;       attributes appropriate to the chain of DDBs being searched.  The
                     0632  1932 ;       created DDB is given the requested DEVNAM and both its primary and
                     0632  1933 ;       secondary UCB chains are initialized to be empty.  The newly created
                     0632  1934 ;       DDB is linked to the end of the DDB and CDDB chains which begin in the
                     0632  1935 ;       input DDB.  Finally, the address of the created DDB is returned.
                     0632  1936 ;
                     0632  1937 ; Inputs:
                     0632  1938 ;
                     0632  1939 ;       R3      address of the requested DEVNAM (a counted ASCII string)
                     0632  1940 ;       R4      address of the beginning DDB for the search
                     0632  1941 ;       R5      UCB address (used solely as a fork block)
                     0632  1942 ;
                     0632  1943 ; Outputs:
                     0632  1944 ;
                     0632  1945 ;       R4      DDB address
                     0632  1946 ;       R5      UCB address (unchanged)
                     0632  1947 ;
                     0632  1948 ;       All other registers are destroyed.
                     0632  1949 ;
                     0632  1950 ; WARNINGS:
                     0632  1951 ;
                     0632  1952 ;       Under some circumstances, this routine will fork using the UCB as a
                     0632  1953 ;       fork block.  Control will be returned to the caller's caller.  Upon
                     0632  1954 ;       entry, 4(SP) must be the return address for the caller's caller.
                     0632  1955 ;--
                     0632  1956
                     0632  1957 DUTU$FIND_DDB::
                     0632  1958
 009C C5 8ED0        0632  1959          POPL    UCB$L_DPC(R5)                   ; Save caller's address.
                     0637  1960
                     0637  1961 5$:      WAIT_FOR_IODB                           ; Obtain write access to the
                     0657  1962                                                  ; I/O database.
                     0657  1963
 00D8 8F    BB       0657  1964          PUSHR   #^M<R3,R4,R6,R7>                ; Save some registers.
 56    53   D0       065B  1965          MOVL    R3, R6                          ; Copy DEVNAM address.
 57    86   9A       065E  1966          MOVZBL  (R6)+, R7                       ; Get size of DEVNAM.
       06   11       0661  1967          BRB     15$                             ; Jump into search loop.
                     0663  1968
   54  38 A4   D0     0663  1969 10$:     MOVL    DDB$L_CONLINK(R4), R4           ; Link to next DDB.
       15   13       0667  1970          BEQL    70$                             ; Branch if no more DDBs.
 14 A4  57   91       0669  1971 15$:     CMPB    R7, DDB$T_NAME(R4)              ; Is DDB DEVNAM the right size?
       F4   12       066D  1972          BNEQ    10$                             ; No, go try next DDB.
15 A4  66   57   29   066F  1973          CMPC3   R7, (R6), DDB$T_NAME+1(R4)      ; Does the name match?
       ED   12       0674  1974          BNEQ    10$                             ; No, go try another DDB.
                     0676  1975
 00C3 8F    BA       0676  1976          POPR    #^M<R0,R1,R6,R7>                ; DEVNAM match found, restore
```

```
                009C D5   17  067A  1977          JMP     @UCB$L_DPC(R5)                  ; saved registers, discard
                                067E  1978                                                ; saved R3 & R4, and return.
                                067E  1979
                                067E  1980                                                ; No DEVNAM match found:
                00D8 8F   BA    067E  1981  70$:  POPR    #^M<R3,R4,R6,R7>                ; Restore saved registers.
              51   44 8F   9A    0682  1982        MOVZBL  #DDB$K_LENGTH, R1               ; Get size of a DDB so that
          00000000'GF   16    0686  1983        JSB     G^EXE$ALONONPAGED               ; a DDB can be allocated.
                46 50   E9    068C  1984        BLBC    R0, 98$                         ; Branch on allocation error.
                   24   BB    068F  1985        PUSHR   #^M<R2,R5>                      ; Save interesting registers.
                7E   53   7D    0691  1986        MOVQ    R3, -(SP)                       ; Save input DEVNAM & DDB addrs.
        62   64   0044 8F   28    0694  1987        MOVC3   #DDB$K_LENGTH, (R4), (R2)       ; Copy original DDB into
                                069A  1988                                                ; newly created DDB.
                   53 8ED0    069A  1989        POPL    R3                              ; Restore requested DEVNAM addr.
                54   83   9A    069D  1990        MOVZBL  (R3)+, R4                       ; Get size of req. DEVNAM.
          52   04 AE   15   C1    06A0  1991        ADDL3   #DDB$T_NAME+1, 4(SP), R2        ; Locate DEVNAM in new DDB.
             62   63   54   28    06A5  1992        MOVC3   R4, (R3), (R2)                  ; Insert requested DEVNAM.
                   38   BA    06A9  1993        POPR    #^M<R3,R4,R5>                   ; Rest. old/new DDB & UCB addrs.
                                06AB  1994
                38 A4   D4    06AB  1995        CLRL    DDB$L_CONLINK(R4)               ; New DDB will end CDDB chain.
                                06AE  1996        ASSUME  DDB$L_UCB EQ <DDB$L_LINK+4>     ; New DDB will also end DDB
                   64   7C    06AE  1997        CLRQ    DDB$L_LINK(R4)                  ; chain.  Null primary UCB link.
                40 A4   D4    06B0  1998        CLRL    DDB$L_2P_UCB(R4)                ; Null secondary UCB link.
                                06B3  1999
                50   53   D0    06B3  2000        MOVL    R3, R0                          ; Init for DDB chain scan.
                     60   D5    06B6  2001  83$:  TSTL    DDB$L_LINK(R0)                  ; End of DDB chain?
                     05   13    06B8  2002        BEQL    84$                             ; Branch if end of chain.
                50   60   D0    06BA  2003        MOVL    DDB$L_LINK(R0), R0              ; Else, link to next DDB.
                     F7   11    06BD  2004        BRB     83$                             ; and loop.
                60   54   D0    06BF  2005  84$:  MOVL    R4, DDB$L_LINK(R0)              ; Put new DDB on end of chain.
                                06C2  2006
                38 A3   D5    06C2  2007  87$:  TSTL    DDB$L_CONLINK(R3)               ; End of CDDB chain?
                     06   13    06C5  2008        BEQL    88$                             ; Branch if end of chain.
             53   38 A3   D0    06C7  2009        MOVL    DDB$L_CONLINK(R3), R3           ; Else, link to next DDB on
                     F5   11    06CB  2010        BRB     87$                             ; CDDB chain and loop.
          38 A3   54   D0    06CD  2011  88$:  MOVL    R4, DDB$L_CONLINK(R3)           ; Put new DDB on end of chain.
                                06D1  2012
                009C D5   17    06D1  2013        JMP     @UCB$L_DPC(R5)                  ; Return to caller.
                                06D5  2014
                                06D5  2015                                                ; Memory allocation failure.
                                06D5  2016  98$:  FORK_WAIT                               ; Wait a little while using
                   FF59   31    06DB  2017        BRW     5$                              ; the UCB as a fork block;
                                06DE  2018                                                ; then try looking for the DDB
                                06DE  2019                                                ; again.  Someone else may
                                06DE  2020                                                ; have created the DDB while
                                06DE  2021                                                ; we were waiting.
```

```
                                        .SBTTL  DUTU$LINK_SEC_UCB - Link UCB to secondary DDB chain
                                ;+
                                ;  DUTU$LINK_SEC_UCB - Link UCB to secondary DDB chain
                                ;
                                ;  Functional Description:
                                ;
                                ;       Search secondary UCB list pointed to by DDB referenced in input UCB and
                                ;       link input UCB into list in ascending unit number order.
                                ;
                                ;  Inputs:
                                ;
                                ;       R2      Address of UCB to be linked
                                ;
                                ;  Implicit Inputs:
                                ;
                                ;       UCB$L_2P_DDB(R2) Address of DDB on which UCB will be hung as a
                                ;                        secondary UCB
                                ;       UCB$W_UNIT(R2) Unit number for UCB
                                ;
                                ;      I/O database available for write access
                                ;
                                ;  Outputs:
                                ;
                                ;       R0      SS$_NORMAL ==> Link operation successful
                                ;               SS$_OPINCOMPL ==> Link operation failed due to presence of UCB
                                ;                       with same unit number
                                ;       R1      Address of UCB following this one in the list
                                ;       R2      Address of this UCB
                                ;       R3      Address of UCB preceding this one in the list
                                ;
                                ;  Implicit Outputs:
                                ;
                                ;       UCB linked in to secondary UCBs chain.
                                ;
                                ;       All non-output registers preserved.
                                ;-
                                DUTU$LINK_SEC_UCB::
                                
51   00A0 C2    00000064 8F    C3  06DE  2062          SUBL3   #<UCB$L_2P_LINK-DDB$L_2P_UCB>, -
                                                              UCB$L_2P_DDB(R2), R1    ; Get address of first UCB link.
                     53    51   D0  06E8  2064 20$:    MOVL    R1, R3                  ; Save address of previous UCB.
                51   00A4 C3   D0  06EB  2065          MOVL    UCB$L_2P_LINK(R3), R1   ; Get address of next UCB.
                           09   13  06F0  2066          BEQL    50$                     ; 0 ==> end-of-list reached; go insert.
              54 A1    54 A2   B1  06F2  2067          CMPW    UCB$W_UNIT(R2), UCB$W_UNIT(R1) ; Compare unit numbers.
                           EF   1A  06F7  2068          BGTRU   20$                     ; If new GT list, continue search.
                           0E   13  06F9  2069          BEQL    90$                     ; If new EQ list, declare error.
                00A4 C2    51   D0  06FB  2070 50$:    MOVL    R1, UCB$L_2P_LINK(R2)   ; Else, link UCB.  Forward link new UCB.
                00A4 C3    52   D0  0700  2071          MOVL    R2, UCB$L_2P_LINK(R3)   ; Forward link previous UCB.
                     50    01   3C  0705  2072          MOVZWL  #SS$_NORMAL, R0         ; Set successful link status
                           05   05  0708  2073          RSB                             ; and return
                                   0709  2074
                     50   02D4 8F   3C  0709  2075 90$:    MOVZWL  #SS$_OPINCOMPL, R0      ; Set link failed status.
                           05   05  070E  2076          RSB                             ; and return.
```

```
                     070F  2078          .SBTTL  DUTU$LINK_UCB2CDDB - Link a UCB into a CDDB chain
                     070F  2079
                     070F  2080 ;++
                     070F  2081 ; DUTU$LINK_UCB2CDDB - Link a UCB into a CDDB chain
                     070F  2082 ;
                     070F  2083 ; Functional Description:
                     070F  2084 ;
                     070F  2085 ;       This routine links the UCB pointed to by R5 into the chain of UCB
                     070F  2086 ;       hanging off the CDDB pointed to by UCB$L_CDDB(R5).  The chain is
                     070F  2087 ;       maintained in ascending MSCP unit number order.
                     070F  2088 ;
                     070F  2089 ; Input Parameters:
                     070F  2090 ;
                     070F  2091 ;       R5      UCB address
                     070F  2092 ;       IPL     IPL$_SCS
                     070F  2093 ;
                     070F  2094 ; Implicit Inputs:
                     070F  2095 ;
                     070F  2096 ;       UCB$L_CDDB(R5)          address of the CDDB anchoring the chain of
                     070F  2097 ;                               UCBs on which the UCB should be hung
                     070F  2098 ;
                     070F  2099 ; Output Parameters:
                     070F  2100 ;
                     070F  2101 ;       IPL     IPL$_SCS
                     070F  2102 ;       R0-R1 destroyed.
                     070F  2103 ;       All other registers preserved.
                     070F  2104 ;
                     070F  2105 ; Implicit Outputs:
                     070F  2106 ;
                     070F  2107 ;       UCB linked into CDDB units chain.
                     070F  2108 ;
                     070F  2109 ; Completion Codes:
                     070F  2110 ;       NONE
                     070F  2111 ;
                     070F  2112 ; Side Effects:
                     070F  2113 ;       NONE
                     070F  2114 ;
                     070F  2115 ;--
                     070F  2116
                     070F  2117 DUTU$LINK_UCB2CDDB::
                     070F  2118
50   00BC C5   FFFFFF84 8F   C1  070F  2119          ADDL3   #<CDDB$L_UCBCHAIN -             ; Initialize previous UCB
                                 0719  2120                   -UCB$L_CDDB_LINK>, -           ; address.
                                 0719  2121                   UCB$L_CDDB(R5), R0
                                 0719  2122
                     51   50   D0  0719  2123 10$:    MOVL    R0, R1                          ; Save previous UCB address.
                50   00C4 C1   D0  071C  2124          MOVL    UCB$L_CDDB_LINK(R1), R0         ; Link to next UCB.
                          09   13  0721  2125          BEQL    80$                             ; Branch if no more UCBs.
           00D4 C0   00D4 C5   B1  0723  2126          CMPW    UCB$W_MSCPUNIT(R5), -           ; Is this MSCP unit no. bigger
                                 072A  2127                   UCB$W_MSCPUNIT(R0)              ; than no. in this chain UCB?
                          ED   1A  072A  2128          BGTRU   10$                             ; Branch if bigger.
                                 072C  2129
                00C4 C5   50   D0  072C  2130 80$:    MOVL    R0, UCB$L_CDDB_LINK(R5)         ; Point new UCB to next UCB.
                00C4 C1   55   D0  0731  2131          MOVL    R5, UCB$L_CDDB_LINK(R1)         ; Point previous UCB at new UCB.
                          05   05  0736  2132          RSB
```

```
                                0737  2134          .SBTTL  DUTU$SEVER_SEC_UCB - Unlink a secondary UCB
                                0737  2135 ;+
                                0737  2136 ;  DUTU$SEVER_SEC_UCB - Unlink a secondary UCB
                                0737  2137 ;
                                0737  2138 ; Functional Description:
                                0737  2139 ;
                                0737  2140 ;       Remove UCB pointed to by R5 from UCB secondary list pointed to by DDB
                                0737  2141 ;       referenced in UCB.
                                0737  2142 ;
                                0737  2143 ;  Inputs:
                                0737  2144 ;
                                0737  2145 ;      R5        Address of UCB to be unlinked
                                0737  2146 ;
                                0737  2147 ;  Implicit Inputs:
                                0737  2148 ;
                                0737  2149 ;       UCB$L_2P_DDB(R5) Address of DDB on which UCB is hung as a secondary UCB
                                0737  2150 ;
                                0737  2151 ;       I/O database available for write access
                                0737  2152 ;
                                0737  2153 ;  Outputs: None.
                                0737  2154 ;
                                0737  2155 ;  Implicit Outputs:
                                0737  2156 ;
                                0737  2157 ;       UCB unlinked from secondary UCBs chain.
                                0737  2158 ;
                                0737  2159 ;       R0 - R1 are destroyed.
                                0737  2160 ;
                                0737  2161 ;-
                                0737  2162
                                0737  2163 DUTU$SEVER_SEC_UCB::
                                0737  2164
50  00A0 C5  00000064 8F  C3  0737  2165          SUBL3   #<UCB$L_2P_LINK-DDB$L_2P_UCB>, -
                                0741  2166                  UCB$L_2P_DDB(R5), R0    ; Get address of first UCB link.
                      51  50  D0  0741  2167 10$:    MOVL    R0, R1                  ; Save address of last UCB.
                  50  00A4 C1  D0  0744  2168          MOVL    UCB$L_2P_LINK(R1), R0   ; Get address of next UCB.
                      55  50  D1  0749  2169          CMPL    R0,R5                   ; Do the UCB addresses match?
                          F3  12  074C  2170          BNEQ    10$                     ; Branch and loop if no match.
          00A4 C1  00A4 C5  D0  074E  2171          MOVL    UCB$L_2P_LINK(R5), -    ; Else, remove UCB from UCB list.
                                0755  2172                  UCB$L_2P_LINK(R1)
                            05  0755  2173          RSB
```

```
                                0756  2175          .SBTTL  DUTU$SEVER_CDDB - Sever UCB from CDDB chain
                                0756  2176
                                0756  2177 ;++
                                0756  2178 ; DUTU$SEVER_CDDB - Sever UCB from CDDB chain
                                0756  2179 ;
                                0756  2180 ; Functional Description:
                                0756  2181 ;
                                0756  2182 ;       This routine severs the UCB pointed to by R5 from the chain of UCB
                                0756  2183 ;       hanging off the CDDB pointed to by UCB$L_CDDB(R5).  The chain is
                                0756  2184 ;       maintained in ascending MSCP unit number order.
                                0756  2185 ;
                                0756  2186 ; Input Parameters:
                                0756  2187 ;
                                0756  2188 ;       R5      UCB address
                                0756  2189 ;       IPL     IPL$_SCS
                                0756  2190 ;
                                0756  2191 ; Implicit Inputs:
                                0756  2192 ;
                                0756  2193 ;       UCB$L_CDDB(R5)          address of the CDDB anchoring the chain of
                                0756  2194 ;                               UCBs from which the UCB should be unlinked
                                0756  2195 ;
                                0756  2196 ; Output Parameters:
                                0756  2197 ;
                                0756  2198 ;       IPL     IPL$_SCS
                                0756  2199 ;       R0-R1 destroyed.
                                0756  2200 ;       All other registers preserved.
                                0756  2201 ;
                                0756  2202 ; Implicit Outputs:
                                0756  2203 ;
                                0756  2204 ;       UCB unlinked from CDDB units chain.
                                0756  2205 ;
                                0756  2206 ; Completion Codes:
                                0756  2207 ;       NONE
                                0756  2208 ;
                                0756  2209 ; Side Effects:
                                0756  2210 ;       NONE
                                0756  2211 ;
                                0756  2212 ;--
                                0756  2213
                                0756  2214 DUTU$SEVER_CDDB::
                                0756  2215
 50   00BC C5    FFFFFF84 8F    C1  0756  2216          ADDL3   #<CDDB$L_UCBCHAIN -          ; Initialize previous UCB
                                0760  2217                    -UCB$L_CDDB_LINK>, -        ; address.
                                0760  2218                    UCB$L_CDDB(R5), R0
                                0760  2219
                          51    50  D0  0760  2220 10$:     MOVL    R0, R1                  ; Save previous UCB address.
                    50    00C4  C1  D0  0763  2221          MOVL    UCB$L_CDDB_LINK(R1), R0 ; Link to next UCB.
                          55    50  D1  0768  2222          CMPL    R0, R5                  ; Is this the severed UCB?
                                F3  12  076B  2223          BNEQ    10$                     ; Branch if not found right UCB.
                                    076D  2224
              00C4 C1   00C4 C5   D0  076D  2225 80$:     MOVL    UCB$L_CDDB_LINK(R5), -  ; Make previous UCB point to
                                    0774  2226                    UCB$L_CDDB_LINK(R1)     ; UCB after severed UCB.
                                05  0774  2227          RSB
```

```
0775  2229          .SBTTL  Default device name and device type tables
0775  2230 ;++
0775  2231 ;
0775  2232 ;       Below the default device name and default device type tables are
0775  2233 ;       built.  DUTU$GET_DEVNAM uses the default device name table to provide
0775  2234 ;       a device name whenever the MSCP media id does not provide one.
0775  2235 ;       DUTU$GET_DEVTYPE uses the default device type table to provide a
0775  2236 ;       device type whenever the MSCP media id cannot be translated into a
0775  2237 ;       device type.  The tables provide for generation of a different device
0775  2238 ;       name and device type for each MSCP controller model.
0775  2239 ;--
0775  2240
0775  2241 ;++
0775  2242 ; MACRO DEF_DD_DT
0775  2243 ;
0775  2244 ; Functional Description:
0775  2245 ;
0775  2246 ;       This macro defines the tables used to "force" a device name and device
0775  2247 ;       type whenever the MSCP media identifier is invalid.  For
0775  2248 ;       DUTU$GET_DEVNAM, entries are placed in the word entry table pointed to
0775  2249 ;       by DUTU$AW_DEF_DEVNAM.  For DUTU$GET_DEVTYPE, entries are placed in
0775  2250 ;       byte entry table pointed to by DUTU$AB_DEF_DEVTYPE.  This macro
0775  2251 ;       automatically constructs patch space in both tables.
0775  2252 ;--
0775  2253          .MACRO  DEF_DD_DT LIST
0775  2254          .MACRO  DEF_ONE ctrlmdl, dd, dt
0775  2255          ASSUME  %LENGTH(dd) EQ 2
0775  2256          .PSECT  $$$220_DEF_DEVNAM_TABLE RD,WRT,EXE,LONG
0775  2257          .=<MSCP$K_CM_'ctrlmdl'*2>
0775  2258          .ASCII  /dd/
0775  2259          .IIF    GT .-$$BIGDD, $$BIGDD=.
0775  2260          .PSECT  $$$220_DEF_DEVTYPE_TABLE RD,WRT,EXE,LONG
0775  2261          .=MSCP$K_CM_'ctrlmdl'
0775  2262          .BYTE   DT$_'dt'
0775  2263          .IIF    GT .-$$BIGDT, $$BIGDT=.
0775  2264          .ENDM   DEF_ONE
0775  2265          .SAVE
0775  2266          .PSECT  $$$220_DEF_DEVNAM_TABLE RD,WRT,EXE,LONG
0775  2267 DUTU$AW_DEF_DEVNAM::
0775  2268 $$BIGDD=.
0775  2269          .PSECT  $$$220_DEF_DEVTYPE_TABLE RD,WRT,EXE,LONG
0775  2270 DUTU$AB_DEF_DEVTYPE::
0775  2271 $$BIGDT=.
0775  2272          .IRP    ITEM, <LIST>
0775  2273          DEF_ONE ITEM
0775  2274          .ENDR
0775  2275          .PSECT  $$$220_DEF_DEVNAM_TABLE RD,WRT,EXE,LONG
0775  2276 .=$$BIGDD
0775  2277          .WORD   0, 0, 0, 0
0775  2278          .PSECT  $$$220_DEF_DEVTYPE_TABLE RD,WRT,EXE,LONG
0775  2279 .=$$BIGDT
0775  2280          .BYTE   0, 0, 0, 0
0775  2281          .RESTORE
0775  2282          .ENDM   DEF_DD_DT
```

```
0775  2284 ; Build the tables
0775  2285
0775  2286         DEF_DD_DT < -
0775  2287                 < HSC50, <DJ>, RA60 >, -
0775  2288                 < UDA50, <DJ>, RA60 >, -
0775  2289                 < RC25,  <DA>, RCF25 >, -
0775  2290                 < EMULA, <DY>, RX02 >, -
0775  2291                 < TU81,  <MU>, TU81 >, -
0775  2292                 < UDA52, <DJ>, RA60 >, -
0775  2293                 < RDRX,  <DU>, RD51 >, -
0775  2294                 >
```

```
0775  2296          .SBTTL  DUTU$GET_DEVNAM - Form new unit device name, DDCn
0775  2297 ;++
0775  2298 ;
0775  2299 ; DUTU$GET_DEVNAM - Form new unit device name, DDCn
0775  2300 ;
0775  2301 ; Functional Description:
0775  2302 ;
0775  2303 ;       Using the MSCP media identification, UCB$L_MEDIA_ID, and MSCP unit
0775  2304 ;       number, UCB$W_MSCPUNIT, fields this routine builds a VMS device name
0775  2305 ;       of the DDCn form.  The "DDC" is stored as a counted ASCII string based
0775  2306 ;       at the address input in R3.  The "n" is stored as an unsigned integer
0775  2307 ;       at UCB$W_UNIT in the input UCB.
0775  2308 ;
0775  2309 ; Inputs:
0775  2310 ;
0775  2311 ;       R3      base address for DDC counted ASCII string
0775  2312 ;       R4      A prototype DDB address (used to determine default controller
0775  2313 ;               letter)
0775  2314 ;       R5      UCB address
0775  2315 ;
0775  2316 ; Implicit Inputs:
0775  2317 ;
0775  2318 ;       UCB$L_CDDB(R5)          CDDB address
0775  2319 ;       UCB$L_MEDIA_ID(R5)      MSCP media identification
0775  2320 ;       UCB$W_MSCPUNIT(R5)      MSCP unit number
0775  2321 ;
0775  2322 ;       For emulated devices the MSCP unit number is assumed to have one of
0775  2323 ;       the following two forms:
0775  2324 ;
0775  2325 ;       15    14      12 11       8 7       4 3       0         with
0775  2326 ;       +-----+----------+----------+---------+---------+       MSCP$M_EU_CTYPE
0775  2327 ;       !SHAD! EU_DESIG ! EU_CTYPE ! EU_SUBC ! EU_SUBU !       equals
0775  2328 ;       +-----+----------+----------+---------+---------+       MSCP$K_EMD_OLD
0775  2329 ;
0775  2330 ;       15    14      12 11       8 7                 0         with
0775  2331 ;       +-----+----------+----------+-------------------+       MSCP$M_EU_CTYPE
0775  2332 ;       !SHAD! EU_DESIG ! EU_CTYPE !       EU_NO       !       not equal to
0775  2333 ;       +-----+----------+----------+-------------------+       MSCP$K_EMD_OLD
0775  2334 ;
0775  2335 ;       SHAD is MSCP$[V,S,M]_SHADOW and is one if the unit is a shadow unit
0775  2336 ;               number.
0775  2337 ;       EU_DESIG is MSCP$[V,S,M]_EU_DESIG and is a number representing the
0775  2338 ;               controller letter used for the device on the node having
0775  2339 ;               direct access (i.e. not MSCP served access) to the device.
0775  2340 ;       EU_CTYPE is MSCP$[V,S,M]_EU_CTYPE and is a constant representing the
0775  2341 ;               type of controller use to access the device on the node having
0775  2342 ;               direct access to the device.  The current choices for
0775  2343 ;               MSCP$M_EU_CTYPE are MSCP$K_EMD_OLD, MSCP$K_EMD_UDA,
0775  2344 ;               MSCP$K_EMD_HSC, MSCP$K_EMD_AZT (for AZTEC), and
0775  2345 ;               MSCP$K_EMD_RDRX.
0775  2346 ;       EU_NO is MSCP$[V,S,M]_EU_NO and is the served unit number for all
0775  2347 ;               values of MSCP$M_EU_CTYPE except MSCK$K_EMD_OLD.
0775  2348 ;       EU_SUBC is MSCP$[V,S,M]_EU_SUBC and is a constant giving a
0775  2349 ;               subcontroller type when MSCP$M_EU_CTYPE is MSCK$K_EMD_OLD.
0775  2350 ;               The current choices for MSCP$M_EU_SUBC are MSCP$K_EMS_RP,
0775  2351 ;               MSCP$K_EMS_RM, MSCP$K_EMS_RK, MSCP$K_EMS_RL, MSCP$K_EMS_RX,
0775  2352 ;               and MSCP$K_EMS_CNSL.
```

```
                     0775  2353 ;       EU_SUBU is MSCP$[V,S,M]_EU_SUBU and is the served unit number when
                     0775  2354 ;               MSCP$M_EU_CTYPE is MSCK$K_EMD_OLD.
                     0775  2355 ;
                     0775  2356 ; Outputs:
                     0775  2357 ;
                     0775  2358 ;       R0 through R2 are destroyed.
                     0775  2359 ;       All other registers are preserved.
                     0775  2360 ;
                     0775  2361 ; Implicit outputs:
                     0775  2362 ;
                     0775  2363 ;       (R3)                    DDC counted ASCII string
                     0775  2364 ;       UCB$W_MSCPUNIT(R5)      VMS unit number
                     0775  2365 ;
                     0775  2366 ;       The DD portion of "DDC" is taken from D0 and D1 in the MSCP media
                     0775  2367 ;       identifier (see the MSCP specification if this does not make sense).
                     0775  2368 ;       The "C" portion of "DDC" is S whenever MSCP$V_SHADOW is set
                     0775  2369 ;       (regardless of serving mechanism).  Otherwise, the "C" portion is A
                     0775  2370 ;       for non-emulated devices and the appropriate MSCP$M_EU_DESIG letter
                     0775  2371 ;       for emulated devices.  The unit number is UCB$W_MSCPUNIT .and.
                     0775  2372 ;       ^C<MSCP$M_SHADOW> for non-emulated devices, UCB$W_MSCPUNIT .and.
                     0775  2373 ;       MSCP$M_EU_NO for emulated devices with MSCP$M_EU_CTYPE not equal to
                     0775  2374 ;       MSCP$K_EMD_OLD, or UCB$W_MSCPUNIT .and. MSCP$M_EU_SUBU for emulated
                     0775  2375 ;       devices with MSCP$M_EU_CTYPE equals MSCP$K_EMD_OLD.
                     0775  2376 ;
                     0775  2377 ;       Note: this routine never uses MSCP$M_EU_DESIG, or MSCP$M_EU_SUBC.  The
                     0775  2378 ;       media identifier is used instead.
                     0775  2379 ;--
                     0775  2380
                     0775  2381 DUTU$GET_DEVNAM:
                     0775  2382
      52   00BC C5   D0   0775  2383          MOVL    UCB$L_CDDB(R5), R2                  ; Get CDDB address.
           63    03   90   077A  2384          MOVB    #3, (R3)                            ; Setup string byte count.
     50     26 A2   9A   077D  2385          MOVZBL  CDDB$B_CNTRLMDL(R2), R0             ; Get MSCP server ctrl. model.
01 A3   0000'CF40   B0   0781  2386          MOVW    W^DUTU$AW_DEF_DEVNAM[R0], 1(R3)     ; Set default "DD" value.
                     0788  2387
50   008C C5   05   1B   EF   0788  2388          EXTZV   #MSCP$V_MTYP_D0, -                  ; Pickup first character of
                     078F  2389                           #MSCP$S_MTYP_D0, -                  ; perfered device mnemonic.
                     078F  2390                           UCB$L_MEDIA_ID(R5), R0
           15   13   078F  2391          BEQL    30$                                 ; Branch if no first char.
01 A3   50     40 8F   81   0791  2392          ADDB3   #^a/A/-1, R0, 1(R3)                 ; Store first character.
50   008C C5   05   16   EF   0797  2393          EXTZV   #MSCP$V_MTYP_D1, -                  ; Pickup second character of
                     079E  2394                           #MSCP$S_MTYP_D1, -                  ; perfered device mnemonic.
                     079E  2395                           UCB$L_MEDIA_ID(R5), R0
           06   13   079E  2396          BEQL    30$                                 ; Branch if no second char.
02 A3   50     40 8F   81   07A0  2397          ADDB3   #^a/A/-1, R0, 2(R3)                 ; Store second character.
                     07A6  2398
     50     14 A4   9A   07A6  2399 30$:     MOVZBL  DDB$T_NAME(R4), R0                  ; Get offset to controller let.
03 A3   14 A440   90   07AA  2400          MOVB    DDB$T_NAME(R4)[R0], 3(R3)           ; Setup default C.
07 00D4 C5   0F   E1   07B0  2401          BBC     #MSCP$V_SHADOW, -                   ; Branch if not a shadow
                     07B6  2402                           UCB$W_MSCPUNIT(R5), 35$             ; unit.
03 A3   53 8F   90   07B6  2403          MOVB    #^a/S7, 3(R3)                       ; Else, set C to be "S",
           15   11   07BB  2404          BRB     50$                                 ; and look no farther.
     26 A2   04   91   07BD  2405 35$:     CMPB    #MSCP$K_CM_EMULA, -                 ; Is this the emulator?
                     07C1  2406                           CDDB$B_CNTRLMDL(R2)
           0F   12   07C1  2407          BNEQ    50$                                 ; Branch if not emulator.
50   00D4 C5   03   0C   EF   07C3  2408          EXTZV   #MSCP$V_EU_DESIG, -                 ; For emulator, get C
                     07CA  2409                           #MSCP$S_EU_DESIG, -                 ; from encoded MSCP unit
```

```
                                07CA  2410                    UCB$W_MSCPUNIT(R5), R0          ; number.
                          06    07CA  2411            BEQL    50$                             ; Branch if no encoded C.
        03 A3   50    40  8F    81    07CC  2412            ADDB3   #^a/A/-1, R0, 3(R3)          ; Else, set encoded C.
                                07D2  2413
     50   00D4 C5    8000  8F    AB    07D2  2414  50$:      BICW3   #MSCP$M_SHADOW, -               ; Get least restrictive
                                07DA  2415                    UCB$W_MSCPUNIT(R5), R0          ; possible unit number.
                26 A2    04    91    07DA  2416            CMPB    #MSCP$K_CM_EMULA, -             ; Is this the emulator?
                                07DE  2417                    CDDB$B_CNTRLMDL(R2)
                          13    12    07DE  2418            BNEQ    59$                             ; Branch if not emulator.
                50    7F00  8F    AA    07E0  2419            BICW    #<MSCP$M_EU_DESIG -             ; Otherwise, clear never
                                07E5  2420                     !MSCP$M_EU_CTYPE>, R0          ; used unit number fields.
                                07E5  2421            ASSUME  MSCP$K_EMD_OLD EQ 0             ; Next, determine whether
        00D4 C5    0F00  8F    B3    07E5  2422            BITW    #MSCP$M_EU_CTYPE, -             ; whether or not EU_NO is
                                07EC  2423                    UCB$W_MSCPUNIT(R5)              ; correct.
                          05    12    07EC  2424            BNEQ    59$                             ; Branch if EU_NO is right.
                50    00F8  8F    AA    07EE  2425            BICW    #MSCP$M_EU_SUBC, R0             ; Else, clear EU_SUBC too.
                  54 A5     50    B0    07F3  2426  59$:      MOVW    R0, UCB$W_UNIT(R5)              ; Establish VMS unit number.
                                07F7  2427
                                05    07F7  2428            RSB                                     ; Return to caller.
```

```
                        .SBTTL  DUTU$GET_DEVTYPE - Translate media-id to a device type
                ;++
                ;
                ; DUTU$GET_DEVTYPE - Translate media-id to a device type
                ;
                ; Functional Description:
                ;
                ;       This routine converts the MSCP media identifier stored in the input
                ;       UCB into a VMS device type which is stored in the input UCB.
                ;
                ; Inputs:
                ;
                ;       R3      UCB address
                ;
                ; Implicit Inputs:
                ;
                ;       UCB$L_MEDIA_ID(R3)      MSCP media identifier value to be converted
                ;       UCB$L_CDDB(R3)          CDDB address
                ;       CDDB$B_CNTRLMDL(cddb)   MSCP controller model
                ;
                ;       MSCP media identifier to VMS device type conversion table in .PSECT
                ;       $$$220_DEVTYPE_TABLE_01
                ;
                ; Outputs:
                ;
                ;       UCB$B_DEVTYPE(R3)       VMS device type
                ;
                ;       R0 is destroyed.
                ;       All other registers are preserved.
                ; --
                
                
                ;
                ; Locate conversion table
                ;
                        .SAVE
                        .PSECT  $$$220_DEVTYPE_TABLE_00 RD,WRT,EXE,LONG
                DUTU$DEVTYPE_TABLE::
                ;
                ; Add some patch space to the conversion table
                ;
                        .PSECT  $$$220_DEVTYPE_TABLE_02 RD,WRT,EXE,BYTE
                        .REPEAT 5
                        .LONG   0
                        .BYTE   0
                        .ENDR
                
                        .RESTORE
```

```
                        07F8  2480 DUTU$GET_DEVTYPE::
                        07F8  2481
    50   0000'CF   9E   07F8  2482          MOVAB   W^DUTU$DEVTYPE_TABLE, R0          ; Get conversion table address.
      41 A3       94   07FD  2483          CLRB    UCB$B_DEVTYPE(R3)                 ; Assume lookup failure.
            06    11   0800  2484          BRB     15$                               ; Jump into table lookup loop.
                       0802  2485
            50    D6   0802  2486 10$:      INCL    R0                                ; Skip unused DEVTYPE value.
            60    D5   0804  2487          TSTL    (R0)                              ; Check for end of table.
            0C    13   0806  2488          BEQL    90$                               ; Branch if end of table.
    80   008C   C3   D1   0808  2489 15$:      CMPL    UCB$L_MEDIA_ID(R3), (R0)+         ; Is this media-id in the table?
            F3    12   080D  2490          BNEQ    10$                               ; Branch if not right id.
      41 A3       60   90   080F  2491          MOVB    (R0), UCB$B_DEVTYPE(R3)           ; Store that DEVTYPE.
                  05   0813  2492          RSB                                       ; Return.
                       0814  2493
                       0814  2494 ; Didn't find media match: use default.
                       0814  2495
    50   00BC   C3   D0   0814  2496 90$:      MOVL    UCB$L_CDDB(R3), R0                ; Get CDDB address.
      50    26   A0   9A   0819  2497          MOVZBL  CDDB$B_CNTRLMDL(R0), R0           ; Get controller model.
41 A3    0000'CF40   90   081D  2498          MOVB    W^DUTU$AB_DEF_DEVTYPE[R0], -      ; Setup default device type.
                       0824  2499                  UCB$B_DEVTYPE(R3)
                  05   0824  2500          RSB                                       ; Return.
```

```
                     0825  2502          .SBTTL  DUTU$INIT_CONN_UCB - Initialize connection dependent UCB fields
                     0825  2503 ;++
                     0825  2504 ;
                     0825  2505 ; DUTU$INIT_CONN_UCB - Initialize connection dependent UCB fields
                     0825  2506 ;
                     0825  2507 ; Functional description:
                     0825  2508 ;
                     0825  2509 ;       This routine initializes those UCB fields which vary with connection.
                     0825  2510 ;       It is called whenever the connection currently in use for a given UCB
                     0825  2511 ;       changes.
                     0825  2512 ;
                     0825  2513 ; Inputs:
                     0825  2514 ;
                     0825  2515 ;       R5      UCB address
                     0825  2516 ;       R3      CDDB address for "current" connection
                     0825  2517 ;
                     0825  2518 ; Implicit Inputs:
                     0825  2519 ;
                     0825  2520 ;       CDDB$L_CDT(R3)  CDT address for "current connection
                     0825  2521 ;       CDDB$L_PDT(R3)  PDT address for "current connection
                     0825  2522 ;
                     0825  2523 ; Outputs:
                     0825  2524 ;
                     0825  2525 ;       All registers preserved.
                     0825  2526 ;
                     0825  2527 ; Implicit Outputs:
                     0825  2528 ;
                     0825  2529 ;       UCB$L_CDT(R5)   CDT address for "current connection
                     0825  2530 ;       UCB$L_PDT(R5)   PDT address for "current connection
                     0825  2531 ;--
                     0825  2532
                     0825  2533 DUTU$INIT_CONN_UCB::
                     0825  2534
00C8 C5   00F4 C3    D0   0825  2535          MOVL    CDDB$L_CDT(R3), UCB$L_CDT(R5)   ; Plant CDT address in UCB.
  0084 C5    14 A3    D0   082C  2536          MOVL    CDDB$L_PDT(R3), UCB$L_PDT(R5)   ; Plant PDT address in UCB.
                     0832  2537
                     05   0832  2538          RSB
```

```
                                                B 6
0833  2540          .SBTTL  DUTU$SETUP_CDP_UCB - Setup remote/local dual pathed device
0833  2541 ;++
0833  2542 ;
0833  2543 ; DUTU$SETUP_CDP_UCB - Setup remote/local dual pathed device
0833  2544 ;
0833  2545 ; Functional Description:
0833  2546 ;
0833  2547 ;       This routine scans the local I/O database for a possible local path to
0833  2548 ;       a MSCP accessed device.  If such a local path is found, both the MSCP
0833  2549 ;       path UCB is altered to point to the local UCB and marked offline.  The
0833  2550 ;       local UCB is also altered to point to the MSCP path UCB.
0833  2551 ;
0833  2552 ; Inputs:
0833  2553 ;
0833  2554 ;       R3      CDDB address
0833  2555 ;       R5      address of a MSCP UCB
0833  2556 ;
0833  2557 ; Implicit Inputs:
0833  2558 ;
0833  2559 ;       UCB$L_DDB(R5)   DDB address for the MSCP device
0833  2560 ;       I/O database locked for scan access
0833  2561 ;
0833  2562 ; Outputs:
0833  2563 ;
0833  2564 ;       R0 - R2 destroyed.
0833  2565 ;       All other preserved.
0833  2566 ;
0833  2567 ; Implicit Outputs:
0833  2568 ;
0833  2569 ;       If a local UCB is found:
0833  2570 ;
0833  2571 ;       UCB$L_2P_ALTUCB(R5)             is set to point to the local UCB
0833  2572 ;       UCB$M_2P and UCB$M_CDP in UCB$L_DEVCHAR2(R5) are set
0833  2573 ;       UCB$L_2P_DDB(R5)                is set to point to the local DDB
0833  2574 ;       UCB$M_ONLINE in UCB$L_STS(R5)   is cleared
0833  2575 ;       UCB$L_2P_ALTUCB(local-UCB)      is set to point to the MSCP UCB
0833  2576 ;       UCB$M_2P in UCB$L_DEVCHAR2(local-UCB) is set
0833  2577 ;       UCB$L_2P_DDB(local-UCB)         is set to point to the MSCP DDB
0833  2578 ;
0833  2579 ; Notes:
0833  2580 ;
0833  2581 ;       This routine has been coded with a seperate local UCB lookup
0833  2582 ;       subroutine in the hope that the routine can eventually be replaced
0833  2583 ;       with a call to an exeuctive routine.
0833  2584 ;--
0833  2585
0833  2586 DUTU$SETUP_CDP_UCB:
0833  2587
26 A3  04   91  0833  2588          CMPB    #MSCP$K_CM_EMULA, -     ; Only disks served by the MSCP server
0837  2589                          CDDB$B_CNTRLMDL(R3)     ; can possibly have a local UCB.  The
       4B   12  0837  2590          BNEQ    99$                     ; local UCB does not count if the it
0839  2591                                                  ; represents UQPORT access to a device,
0839  2592                                                  ; since such devices are already
0839  2593                                                  ; serviced by the class driver.
0839  2594
   01D8 8F   BB  0839  2595          PUSHR   #^M<R3,R4,R6,R7,R8>     ; Save some registers.
50    28 A5   D0  083D  2596          MOVL    UCB$L_DDB(R5), R0       ; Get DDB address.
```

```
                58   3C A0    D0  0841  2597          MOVL    DDB$L_ALLOCLS(R0), R8   ; Get allocation class.
                             39  13  0845  2598          BEQL    90$                     ; Branch if allocation class not unique.
                57   14 A0    9E  0847  2599          MOVAB   DDB$T_NAME(R0), R7      ; Get "DDC" address.
                56   54 A5    3C  084B  2600          MOVZWL  UCB$W_UNIT(R5), R6      ; Get unit number.
                             34  10  084F  2601          BSBB    LOOKUP_LOCAL_UCB        ; Attempt to find a local UCB.
                             50  D5  0851  2602          TSTL    R0                      ; Was a local UCB found?
                             2B  13  0853  2603          BEQL    90$                     ; Branch if no local UCB found.
             26 3C A0        05  E0  0855  2604          BBS     #DEV$V_MSCP, -          ; Branch if the local UCB found
                                 085A  2605                  UCB$L_DEVCHAR2(R0), 90$ ; is a class driver UCB.
                                 085A  2606
              00A8 C0    55   D0  085A  2607          MOVL    R5, UCB$L_2P_ALTUCB(R0) ; Else point UCB's at each other.
              00A8 C5    50   D0  085F  2608          MOVL    R0, UCB$L_2P_ALTUCB(R5)
                3C A5    18   C8  0864  2609          BISL    #<DEV$M_CDP -           ; Mark the MSCP UCB to show that there
                                 0868  2610                   !DEV$M_2P>, -          ; is a local path to the device and
                                 0868  2611                  UCB$L_DEVCHAR2(R5)      ; it represents a dual pathed device.
                3C A0    10   C8  0868  2612          BISL    #DEV$M_2P, -            ; Mark the local UCB to show that it
                                 086C  2613                  UCB$L_DEVCHAR2(R0)      ; represents a dual pathed device.
            00A0 C5    28 A0    D0  086C  2614          MOVL    UCB$L_DDB(R0), -        ; Setup alternate path DDB pointer
                                 0872  2615                  UCB$L_2P_DDB(R5)        ; in MSCP UCB.
            00A0 C0    28 A5    D0  0872  2616          MOVL    UCB$L_DDB(R5), -        ; Setup alternate path DDB pointer
                                 0878  2617                  UCB$L_2P_DDB(R0)        ; in local UCB.
                64 A5    10   CA  0878  2618          BICL    #UCB$M_ONLINE, -        ; Mark the MSCP served UCB as being
                                 087C  2619                  UCB$L_STS(R5)           ; unusable.
                                 087C  2620          ASSUME  DEV$V_AVL GE 16         ; Also make allocation and channel
                3A A5    04   8A  087C  2621          BICB    #<DEV$M_AVL @ -16>, -   ; assignments impossible on the
                                 0880  2622                  UCB$L_DEVCHAR+2(R5)     ; MSCP server UCB.
                                 0880  2623
                    01D8 8F      BA  0880  2624 90$:   POPR    #^M<R3,R4,R6,R7,R8>     ; Restore registers.
                                 0884  2625
                                 05  0884  2626 99$:   RSB
```

```
                                0885  2628 ;++
                                0885  2629 ; LOOKUP_LOCAL_UCB - Search for a local UCB
                                0885  2630 ;
                                0885  2631 ; Functional Description:
                                0885  2632 ;
                                0885  2633 ;       This routine searches the local I/O database for a UCB whose
                                0885  2634 ;       allocation class, and unit name, DDCn form, match those input.
                                0885  2635 ;
                                0885  2636 ; Inputs:
                                0885  2637 ;
                                0885  2638 ;       R6      desired unit number
                                0885  2639 ;       R7      address of desired DDC counted ASCII string
                                0885  2640 ;       R8      desired allocation class (presumable not zero)
                                0885  2641 ;
                                0885  2642 ; Outputs:
                                0885  2643 ;
                                0885  2644 ;       R0      address of a UCB meeting the criteria (or zero)
                                0885  2645 ;
                                0885  2646 ;       R0 through R4 are destroyed.
                                0885  2647 ;       All other registers are preserved.
                                0885  2648 ;
                                0885  2649 ; Notes:
                                0885  2650 ;
                                0885  2651 ;       Hopefully, the fullness of time will allow this routine to be
                                0885  2652 ;       replaced by a similar routine in the executive.
                                0885  2653 ;--
                                0885  2654
                                0885  2655 LOOKUP_LOCAL_UCB:
                                0885  2656
                                0885  2657          ASSUME  DDB$L_LINK EQ 0
      54   00000000'GF   DE     0885  2658          MOVAL   G^IOC$GL_DEVLIST, R4    ; Get initial DDB address.
                           088C  2659
           54   64   D0   088C  2660 10$:    MOVL    DDB$L_LINK(R4), R4      ; Get next DDB.
                     2A   13   088F  2661          BEQL    90$                     ; Branch if no more DDBs.
           3C A4     58   D1   0891  2662          CMPL    R8, DDB$L_ALLOCLS(R4)   ; Right allocation class?
                     F5   12   0895  2663          BNEQ    10$                     ; Branch if wrong allocation class.
           14 A4     67   91   0897  2664          CMPB    (R7), DDB$T_NAME(R4)    ; Right "DDC" string size?
                     EF   12   089B  2665          BNEQ    10$                     ; Branch if wrong "DDC" string size.
                51   67   9A   089D  2666          MOVZBL  (R7), R1                ; Get "DDC" string size.
      15 A4    01 A7   51   29   08A0  2667          CMPC3   R1, 1(R7), -            ; Right "DDC" name?
                                08A6  2668                  DDB$T_NAME+1(R4)
                     E4   12   08A6  2669          BNEQ    10$                     ; Branch if wrong "DDC" name.
                           08A8  2670
           50    D4 A4   9E   08A8  2671          MOVAB   <DDB$L_UCB -            ; Initialize UCB address.
                                08AC  2672                  -UCB$L_LINK>(R4), R0
                           08AC  2673
           50    30 A0   D0   08AC  2674 20$:    MOVL    UCB$L_LINK(R0), R0      ; Get next UCB address.
                     09   13   08B0  2675          BEQL    90$                     ; Branch if no more UCBs.
           54 A0     56   B1   08B2  2676          CMPW    R6, UCB$W_UNIT(R0)      ; Is this the right unit?
                     F4   1A   08B6  2677          BGTRU   20$                     ; Branch if more units to check.
                     01   12   08B8  2678          BNEQ    90$                     ; Branch if not right unit.
                          05   08BA  2679          RSB                             ; Return if unit found.
                           08BB  2680
                     50   D4   08BB  2681 90$:    CLRL    R0                      ; Indicate no UCB found.
                          05   08BD  2682          RSB                             ; Exit
```

```
                        08BE  2684          .SBTTL  ----- CANCEL ROUTINES -----
                        08BE  2685          .SBTTL  Cancel-CDRP Definitions
                        08BE  2686 ;++
                        08BE  2687 ;
                        08BE  2688 ;       The special IRP/CDRP pair (the cancel-CDRP) allocated to represent a
                        08BE  2689 ;       cancel request is used only by the class driver.  It should never be
                        08BE  2690 ;       seen by any of the standard VMS I/O processing routines.  Therefore,
                        08BE  2691 ;       several IRP fields in the cancel-CDRP are resued to represent elements
                        08BE  2692 ;       of the cancel request.
                        08BE  2693 ;
                        08BE  2694 ;       Canceled IRP/CDRP pairs requiring an ABORT command also have a field
                        08BE  2695 ;       reused.  It is the I/O queue forward link.  These IRP/CDRPs will be
                        08BE  2696 ;       returned to the control of cancel processing before that field is need
                        08BE  2697 ;       for another purpose.
                        08BE  2698 ;
                        08BE  2699 ;       These field redefinitions are accomplished below.  These definitions
                        08BE  2700 ;       apply only to the cancel routines which follow.
                        08BE  2701 ;--
                        08BE  2702
                        08BE  2703 ; Define CDDB cancel queue links
                        08BE  2704
                        08BE  2705 ASSUME CDRP$L_IOQBL EQ <CDRP$L_IOQFL + 4>
                        08BE  2706 ASSUME IRP$L_IOQBL EQ <IRP$L_IOQFL + 4>
FFFFFFA0                08BE  2707 CDRP$L_CANIOQFL = CDRP$L_IOQFL
FFFFFFA4                08BE  2708 CDRP$L_CANIOQBL = CDRP$L_IOQBL
00000000                08BE  2709 IRP$L_CANIOQFL = IRP$L_IOQFL
00000004                08BE  2710 IRP$L_CANIOQBL = IRP$L_IOQBL
                        08BE  2711
                        08BE  2712 ; Define send an ABORT command queue header.
                        08BE  2713
                        08BE  2714 ASSUME CDRP$L_OBCNT EQ <CDRP$L_ABCNT + 4>
                        08BE  2715 ASSUME IRP$L_OBCNT EQ <IRP$L_ABCNT + 4>
FFFFFFE0                08BE  2716 CDRP$L_SNDABTQFL = CDRP$L_ABCNT
FFFFFFE4                08BE  2717 CDRP$L_SNDABTQBL = CDRP$L_OBCNT
00000040                08BE  2718 IRP$L_SNDABTQFL = IRP$L_ABCNT
00000044                08BE  2719 IRP$L_SNDABTQBL = IRP$L_OBCNT
                        08BE  2720
                        08BE  2721 ; Define ABORT sent queue header.
                        08BE  2722
                        08BE  2723 ASSUME CDRP$L_IOST2 EQ <CDRP$L_IOST1 + 4>
                        08BE  2724 ASSUME IRP$L_IOST2 EQ <IRP$L_IOST1 + 4>
FFFFFFD8                08BE  2725 CDRP$L_ABTDQFL = CDRP$L_IOST1
FFFFFFDC                08BE  2726 CDRP$L_ABTDQBL = CDRP$L_IOST2
00000038                08BE  2727 IRP$L_ABTDQFL = IRP$L_IOST1
0000003C                08BE  2728 IRP$L_ABTDQBL = IRP$L_IOST2
                        08BE  2729
                        08BE  2730 ; Define place to save RSPID of canceled MSCP command.
FFFFFFF0                08BE  2731 CDRP$L_CAN_RSPID = CDRP$L_SEQNUM
00000050                08BE  2732 IRP$L_CAN_RSPID = IRP$L_SEQNUM
                        08BE  2733
                        08BE  2734 ; Define CANIO CDRP backpointer (for canceled IRP/CDRPs only).
FFFFFFA0                08BE  2735 CDRP$L_CANIOCDRP = CDRP$L_IOQFL
00000000                08BE  2736 IRP$L_CANIOCDRP = IRP$L_IOQFL
```

```
08BE  2738          .SBTTL  DUTU$CANCEL - Cancel I/O Routine for class drivers
08BE  2739 ;++
08BE  2740 ;
08BE  2741 ; DUTU$CANCEL - Cancel I/O Routine for class drivers
08BE  2742 ;
08BE  2743 ; Functional Description:
08BE  2744 ;
08BE  2745 ;       This is the driver cancel I/O routine for the disk and tape class
08BE  2746 ;       drivers.  It is called directly from the $CANCEL system service (due
08BE  2747 ;       to the vector information in the class driver DDT.
08BE  2748 ;
08BE  2749 ;       Cancel processing interacts with other significant class driver
08BE  2750 ;       operations including; device failover, reconnection processing, and
08BE  2751 ;       host initiated bad block replacement (for disks only).  Generally
08BE  2752 ;       speaking, these interactions are deliniated in this routine and the
08BE  2753 ;       routines which follow it.  However, the bad block replacement code is
08BE  2754 ;       in DUHIRT (due to the limited nature of its relivance).
08BE  2755 ;
08BE  2756 ;       The multi-threaded nature of the class drivers significantly
08BE  2757 ;       complicates processing for the cancel function.  The requests to be
08BE  2758 ;       canceled must be located in any one of several different places.
08BE  2759 ;       Several different requests can be currently active.  Several requests
08BE  2760 ;       can be stalled waiting for resources.
08BE  2761 ;
08BE  2762 ;       As requests to be canceled are located they will either have an
08BE  2763 ;       associated MSCP request pending in the remote server, or for any one
08BE  2764 ;       of several reasons, they will not have associated MSCP server
08BE  2765 ;       requests.  When an associated MSCP server request exists, an MSCP
08BE  2766 ;       ABORT command must be sent to the remote server.  Otherwise, the
08BE  2767 ;       request can be completed immediately.
08BE  2768 ;
08BE  2769 ;       Before actually processing the cancel request, a check is maded to
08BE  2770 ;       determine whether or not a similar cancel request is still being
08BE  2771 ;       processed.  If a similar request is found, the routine exits without
08BE  2772 ;       performing any processing; the cancel request is redundant and can be
08BE  2773 ;       ignored.  Otherwise, further cancel processing is performed.
08BE  2774 ;
08BE  2775 ;       Next a IRP/CDRP pair is allocated to represent the cancel request.
08BE  2776 ;       Several fields in the IRP/CDRP are initialized, including the
08BE  2777 ;       CDRP$V_CANIO flag in CDRP$L_DUTUFLAGS which marks this IRP/CDRP as a
08BE  2778 ;       special IRP/CDRP representing a pending cancel request.  Henceforth,
08BE  2779 ;       this special IRP/CDRP will be refered to as the cancel-CDRP.  The
08BE  2780 ;       cancel-CDRP is linked to the CDDB cancel queue (CDDB$L_CANCLQFL) so
08BE  2781 ;       that it can be located easily.  Failure to allocate a cancel-CDRP
08BE  2782 ;       results in immediate return to the $CANCEL system service without
08BE  2783 ;       having accomplished anything.
08BE  2784 ;
08BE  2785 ;       Now, all the little nooks and crannies in which class driver I/O
08BE  2786 ;       requests can be stashed are searched for IRP/CDRPs which need to be
08BE  2787 ;       canceled.  All IRP/CDRP pairs located are tested against the cancel
08BE  2788 ;       criteria.  An attempt is made to locate IRP/CDRPs holding no SCS
08BE  2789 ;       resources first, IRP/CDRPs waiting for some SCS resources next, and
08BE  2790 ;       IRP/CDRPs with assoicated MSCP requests last.
08BE  2791 ;
08BE  2792 ;       Those IRP/CDRPs which do not have an associated MSCP request are
08BE  2793 ;       immediately sent to I/O post processing.  Those IRP/CDRPs which do
08BE  2794 ;       have associated MSCP requests are marked as canceled (the CDRP$V_CAND
```

```
;       bit is set in CDDB$L_DUTUFLAGS) and queued to the send an ABORT
;       message queue hanging off of the cancel-CDRP (CDRP$L_SNDABTQFL).
;
;       N.B. no MSCP ABORT commands are sent until ALL IRP/CDRPs meeting the
;       cancel criteria are located and processed.
;
;       At this point, it is possible to adjust the wait counter based upon
;       the required adjustment factor accumulated as the IRP/CDRPs lookup and
;       processing performed above.  After adjusting the wait count, activity
;       on the device -- which was stalled due to an elevated wait count -- is
;       resumed.
;
;       Since all the IRP/CDRP pairs meeting the cancel criteria have been
;       located, it now is possible to issue the needed MSCP ABORT commands
;       in an asynchronous fork thread and return to the $CANCEL system
;       service.
;
;       Interactions with other class driver operations:
;
;       The major complication with other operations has to do with reforming
;       broken connections to the MSCP server.  Four distinct situations might
;       arise:
;
;           o   the $CANCEL system service might be invoked during the middle
;               of a reconnection attempt.  In this case, cancelable requests
;               will be found on the restart queue (CDDB$L_RSTRTQFL) and
;               terminated immediately, as they do not have an associated MSCP
;               request.  A single cancelable request may be found to have an
;               associated MSCP request, as a result of single step mode.  It
;               must have an ABORT command sent to the remote MSCP server.
;
;           o   the connection might fail during the lookup of IRP/CDRPs
;               requiring cancelation (i.e. before the ABORT commands can be
;               sent).  This condition is recognized by a failure to allocate
;               a message buffer to send the ABORT.  All the IRP/CDRPs to be
;               canceled have already been located and removed from normal
;               class driver work and SCS resource wait queues.  Therefore,
;               they will not be automatically restarted; they only need to be
;               queued for I/O post processing.  Then, the cancel operation
;               can simply be terminated.
;
;           o   the connection might fail while and ABORT command is
;               outstanding.  This condition is detected by DUTU$INSERT_RESTRTQ
;               when it tries to insert a cancel-CDRP on the restart queue.
;               Instead of queueing the cancel-CDRP, all remaining IRP/CDRPs
;               to be canceled are queued for I/O post processing and the
;               cancel operation is terminated.
;
;           o   the connection might fail after all the ABORT have completed
;               but before all the canceled I/O requests have terminated.
;               This is handled by reconnection processing.  At an appropriate
;               time, all canceled IRP/CDRPs are queued for I/O post
;               processing and the cancel operation is terminated.
;
;       There also is an interaction with device failover.  Because the
;       presence of incomplete cancel operations signals incomplete but soon
;       to be completed activity on a device and because it is not convenient
```

```
                     08BE  2852 ;       to move that activity to a new path, device failover is not permitted
                     08BE  2853 ;       when incomplete cancel operations are detected for the device.
                     08BE  2854 ;
                     08BE  2855 ;       The final, and perhaps most important, interaction with other class
                     08BE  2856 ;       driver operations occurs in MSCP end-packet processing.  The common
                     08BE  2857 ;       MSCP end-packet processing routine tests the CDRP$V_CAND bit
                     08BE  2858 ;       associated with each incomming MSCP end-packet.  If this bit is set,
                     08BE  2859 ;       DUTU$END_CAND_REQ is called, to test for possible completion of the
                     08BE  2860 ;       pending cancel operation.
                     08BE  2861 ;
                     08BE  2862 ; Inputs:
                     08BE  2863 ;
                     08BE  2864 ;       R2      Channel index number on which cancel is to be performed
                     08BE  2865 ;       R4      Process PCB address
                     08BE  2866 ;       R5      Device UCB address
                     08BE  2867 ;
                     08BE  2868 ;       IPL     UCB$B_FIPL
                     08BE  2869 ;
                     08BE  2870 ; Outputs:
                     08BE  2871 ;
                     08BE  2872 ;       R0 through R3 are destroyed.
                     08BE  2873 ;       All other registers are preserved.
                     08BE  2874 ;--
                     08BE  2875
                     08BE  2876 DUTU$CANCEL::
                     08BE  2877
     00E0    30      08BE  2878         BSBW    DUTU$CHECK_NOCANCEL     ; Check for similar cancel request.
  01 50      E8      08C1  2879         BLBS    R0, 10$                 ; Branch if no similar cancel request
                     08C4  2880                                         ; is in progress.
             05      08C4  2881 9$:     RSB                             ; Else, exit doing nothing.
                     08C5  2882
     010C    30      08C5  2883 10$:    BSBW    DUTU$BUILD_CANIO_CDRP   ; Build cancel-CDRP.
   F9 50     E9      08C8  2884         BLBC    R0, 9$                  ; Branch if CDRP build failed.
                     08CB  2885
        53   55      D0    08CB  2886         MOVL    R5, R3                  ; Move UCB address.
     55   60 A1      9E    08CE  2887         MOVAB   IRP$L_FQFL(R1), R5      ; Get cancel-CDRP address.
                     08D2  2888
                     08D2  2889
                     08D2  2890 ;
                     08D2  2891 ; Search for IRP/CDRP pairs meeting the cancel criteria.
                     08D2  2892 ;
                     08D2  2893
                     08D2  2894         ; Search through the restart queue for IRP/CDRPs to cancel.
                     08D2  2895
51   00BC C3    3C    C1    08D2  2896         ADDL3   #CDDB$L_RSTRTQFL, -      ; Get restart queue header address.
                     08D8  2897                         UCB$L_CDDB(R3), R1
        52   51      D0    08D8  2898         MOVL    R1, R2                  ; Copy header address.
                     08DB  2899
        52   62      D0    08DB  2900 110$:   MOVL    CDRP$L_FQFL(R2), R2     ; Link to next restart CDRP.
        51   52      D1    08DE  2901         CMPL    R2,R1                   ; End of queue?
             18      13    08E1  2902         BEQL    130$                    ; Branch if end of restart queue.
                     08E3  2903         IFNOCANCEL cdrp=(R2), then=110$ ; Branch if don't cancel this CDRP.
        50   62      0F    08E9  2904         REMQUE  (R2), R0                ; Dequeue CDRP & get address in R0.
                     08EC  2905         POST_CDRP status=SS$_CANCEL     ; Insert packet in IOPOST queue.
             E0      11    08F9  2906         BRB     110$
                     08FB  2907
                     08FB  2908 130$:   ; Cancel all requests waiting for (or using) the HIRT.
```

```
                                08FB  2909
                                08FB  2910          .WEAK   DU$CANCEL_FROM_HIRT
50  00000000'GF   9E  08FB  2911          MOVAB   G^DU$CANCEL_FROM_HIRT, R0  ; Get HIRT cancel routine address.
              02   13  0902  2912          BEQL    150$                        ; Branch if no HIRT routine to call.
              60   16  0904  2913          JSB     (R0)                        ; Else, call cancel from HIRT.
                                0906  2914
                                0906  2915 150$:   ; Search the RDT wait queue and the RDT itself.
                                0906  2916
    53    24 A5   D0  0906  2917          MOVL    CDRP$L_CDT(R5), R3          ; Get CDT address for SCS searches.
       51   55   D0  090A  2918          MOVL    R5, R1                      ; Copy CANIO CDRP addr. for SCS search.
                                090D  2919
                                090D  2920          SCAN_RSPID_WAIT -           ; Scan RDT wait queue.
                                090D  2921                  action = DUTU$CANCEL_RDTWAIT
                                091A  2922          SCAN_RDT -                  ; Scan RDT.
                                091A  2923                  action = DUTU$CANCEL_RDT
       55   51   D0  0927  2924          MOVL    R1, R5                      ; Restore CANIO CDRP address to R5.
                                092A  2925
                                092A  2926 ;
                                092A  2927 ; Update wait count, and if necessary, unstall UCB.
                                092A  2928 ;
                                092A  2929
    50    44 A5   B0  092A  2930          MOVW    CDRP$W_DUTUCNTR(R5), R0 ; Get number of decrements to do.
              14   13  092E  2931          BEQL    200$                        ; Branch if no decrements required.
          7E   54   7D  0930  2932          MOVQ    R4, -(SP)                   ; Save registers of interest.
    55     BC A5   D0  0933  2933          MOVL    CDRP$L_UCB(R5), R5          ; Get UCB to unstall.
    56  A5    50   A2  0937  2934          SUBW    R0, UCB$W_RWAITCNT(R5)      ; Decrement wait count.
  00000000'GF   16   93B  2935          JSB     G^SCS$UNSTALLUCB            ; Start up any stalled requests.
          54   8E   7D  0941  2936          MOVQ    (SP)+, R4                   ; Restore registers.
                                0944  2937
                                0944  2938 ;
                                0944  2939 ; If MSCP ABORT commands are required, start a fork thread to send them.
                                0944  2940 ; Otherwise, simply end the cancel request now.
                                0944  2941 ; In any case, return to the $CANCEL system service.
                                0944  2942 ;
                                0944  2943
50     E0 A5    9E  0944  2944 200$:   MOVAB   CDRP$L_SNDABTQFL(R5), R0; Get send-abort queue header.
    60    50   D1  0948  2945          CMPL    R0, (R0)                    ; Test for an empty queue.
             0A   12  094B  2946          BNEQ    250$                        ; Branch if queue not empty.
       BC A5   DD  094D  2947          PUSHL   CDRP$L_UCB(R5)              ; Else, save UCB address.
        01A9   30  0950  2948          BSBW    DUTU$END_CANCEL             ; End cancel operation now.
          55 8ED0  0953  2949          POPL    R5                          ; Restore UCB address.
             05   0956  2950          RSB                                 ; Return to cancel system service.
                                0957  2951
                                0957  2952 250$:   CREATE_FORK -                 ; Create fork thread to
                                0957  2953                  SEND_ABORTS, -       ; send the needed ABORT commands.
                                0957  2954                  frkblk = (R5), -
                                0957  2955                  mask = <^M<R4>>
                                095E  2956
55     BC A5    D0  095E  2957          MOVL    CDRP$L_UCB(R5), R5          ; Restore UCB address in R5.
             05   0962  2958          RSB                                 ; Return to $CANCEL system service.
```

```
                     0963  2960          .SBTTL  SEND_ABORTS - Send ABORT commands for active MSCP requests
                     0963  2961 ;++
                     0963  2962 ;
                     0963  2963 ; SEND_ABORTS - Send ABORT commands for active MSCP requests
                     0963  2964 ;
                     0963  2965 ; Functional Description:
                     0963  2966 ;
                     0963  2967 ;       This independent fork thread sends MSCP ABORT commands for every CDRP
                     0963  2968 ;       on a cancel-CDRP SNDABT queue.  The cancel-CDRP is used as a fork
                     0963  2969 ;       block for these operations.  Each CDRP queued to CDRP$L_SNDABTQFL is
                     0963  2970 ;       removed from that queue and queued to CDRP$L_ABTDQBL, the ABORT sent
                     0963  2971 ;       queue.  Then an MSCP ABORT command is sent to the remote server.
                     0963  2972 ;       This proceeds until CDRP$L_SNDABTQFL is empty.
                     0963  2973 ;
                     0963  2974 ; Inputs:
                     0963  2975 ;
                     0963  2976 ;       R4      PDT address
                     0963  2977 ;       R5      cancel-CDRP address
                     0963  2978 ;
                     0963  2979 ; Outputs:
                     0963  2980 ;
                     0963  2981 ;       R0 through R5 destroyed.
                     0963  2982 ;       All other registers preserved.
                     0963  2983 ;--
                     0963  2984
                     0963  2985 SEND_ABORTS:
                     0963  2986
     53    BC A5    D0  0963  2987          MOVL    CDRP$L_UCB(R5), R3              ; Get UCB address.
54   0084 C3    D0  0967  2988          MOVL    UCB$L_PDT(R3), R4               ; Get PDT address.
                     096C  2989
     51    E0 B5    0F  096C  2990 10$:    REMQUE  @CDRP$L_SNDABTQFL(R5), R1       ; Get CDRP to abort.
           29    1D  0970  2991          BVS     90$                             ; Branch if no more CDRPs.
                     0972  2992
        DC B5    61  0E  0972  2993          INSQUE  (R1), @CDRP$L_ABTDQBL(R5)       ; Queue "aborted" CDRP.
                     0976  2994
   F0 A5    20 A1    D0  0976  2995          MOVL    CDRP$L_RSPID(R1), -             ; Save RSPID to abort.
                     097B  2996                  CDRP$L_CAN_RSPID(R5)
                     097B  2997
                     097B  2998          ALLOC_RSPID                             ; Allocate RSPID for ABORT cmd.
                     0981  2999          ALLOC_MSG_BUF                           ; Allocate a message buffer too.
           17 50    E9  0984  3000          BLBC    R0, 900$                        ; Branch if connection broke.
                     0987  3001          INIT_MSCP_MSG ucb=(R3)                  ; Initialize ABORT message.
      08 A2    01    90  098A  3002          MOVB    #MSCP$K_OP_ABORT, -             ; Set ABORT MSCP opcode.
                     098E  3003                  MSCP$B_OPCODE(R2)
   0C A2    F0 A5    D0  098E  3004          MOVL    CDRP$L_CAN_RSPID(R5), -         ; Set RSPID to abort.
                     0993  3005                  MSCP$L_OUT_REF(R2)
                     0993  3006          SEND_MSCP_MSG                           ; Send ABORT command.
            033A    30  0996  3007          BSBW    DUTU$DEALLOC_RSPID_MSG          ; Deallocate RSPID & msg. buf.
                     0999  3008
              D1    11  0999  3009          BRB     10$                             ; Repeat until no more CDRPs
                     099B  3010                                                  ; to be canceled.
                     099B  3011
                     099B  3012 90$:    ; All needed ABORT commands have been sent.
                     099B  3013
              65    7C  099B  3014          CLRQ    CDRP$L_FQFL(R5)                 ; Signal cancel-CDRP not queued.
                 05    099D  3015          RSB                                     ; Terminate fork thread.
                     099E  3016
```

```
                     099E  3017 900$:   ; Connention to the remote server broke.  Complete this cancel
                     099E  3018         ; operation now.  No more abort commands need be sent.
                     099E  3019
015B   31            099E  3020         BRW     DUTU$END_CANCEL                 ; Complete cancel operation
                     09A1  3021                                                 ; and terminate fork thread.
```

```
                     09A1  3023          .SBTTL  DUTU$CHECK_NOCANCEL - Check for no similar cancel requests
                     09A1  3024 ;++
                     09A1  3025 ;
                     09A1  3026 ; DUTU$CHECK_NOCANCEL - Check for no similar cancel requests
                     09A1  3027 ;
                     09A1  3028 ; Functional Description:
                     09A1  3029 ;
                     09A1  3030 ;       This routine scans the CDDB in-progress cancel operations queue for
                     09A1  3031 ;       the primary CDDB of the input device looking for cancel operations
                     09A1  3032 ;       similar to that described by the other input registers.  If a no
                     09A1  3033 ;       similar cancel operation is found, success status is returned.  If as
                     09A1  3034 ;       many as one similar cancel operation is found, failure status is
                     09A1  3035 ;       returned.  The input criteria may describe an entire cancel operation
                     09A1  3036 ;       or just the relivant UCB.
                     09A1  3037 ;
                     09A1  3038 ; Inputs:
                     09A1  3039 ;
                     09A1  3040 ;       R2      Channel index for cancel operation
                     09A1  3041 ;       R4      PCB address (or zero if PID and channel index tests are not to
                     09A1  3042 ;                   be done)
                     09A1  3043 ;       R5      UCB address
                     09A1  3044 ;
                     09A1  3045 ; Implicit inputs:
                     09A1  3046 ;
                     09A1  3047 ;       UCB$L_CDDB(R5)  CDDB address
                     09A1  3048 ;
                     09A1  3049 ; Outputs:
                     09A1  3050 ;
                     09A1  3051 ;       R0      LBS ==> no similar cancel operation found
                     09A1  3052 ;               LBC ==> similar cancel operation found
                     09A1  3053 ;       R1      is destroyed.
                     09A1  3054 ;       All other registers are preserved.
                     09A1  3055 ;--
                     09A1  3056
                     09A1  3057 DUTU$CHECK_NOCANCEL:
                     09A1  3058
                     09A1  3059          ASSUME  IRP$L_CANIOQFL EQ IRP$L_IOQFL
                     09A1  3060          ASSUME  IRP$L_IOQFL EQ 0
                     09A1  3061
50   00BC C5   000000B0 8F   C1  09A1  3062          ADDL3   #CDDB$L_CANCLQFL, -                  ; Get CDDB cancel queue header
                                 09AB  3063                  UCB$L_CDDB(R5), R0                  ; address.
                        51   50  D0  09AB  3064          MOVL    R0, R1                          ; Copy that address.
                                 09AE  3065
                        50   60  D0  09AE  3066 10$:      MOVL    IRP$L_CANIOQFL(R0), R0          ; Link to next CANIO CDRP.
                        51   50  D1  09B1  3067          CMPL    R0, R1                          ; Is this end of queue.
                             1A  13  09B4  3068          BEQL    100$                            ; Branch if end of queue.
                     1C A0   55  D1  09B6  3069          CMPL    R5, IRP$L_UCB(R0)               ; Is this the right UCB?
                             F2  12  09BA  3070          BNEQ    10$                             ; Branch if wrong UCB.
                             54  D5  09BC  3071          TSTL    R4                              ; Are full tests needed?
                             0D  13  09BE  3072          BEQL    90$                             ; Branch if full tests unneeded.
                  0C A0    60  A4  D1  09C0  3073          CMPL    PCB$L_PID(R4), IRP$L_PID(R0)    ; Is this the right PID?
                             E7  12  09C5  3074          BNEQ    10$                             ; Branch if wrong PID.
                     28 A0   52  B1  09C7  3075          CMPW    R2, IRP$W_CHAN(R0)              ; Is this the right channel?
                             E1  12  09CB  3076          BNEQ    10$                             ; Branch if wrong channel index.
                                 09CD  3077
                                 09CD  3078 90$:      ; Oops.  Found a similar cancel request.
                             50  D4  09CD  3079          CLRL    R0                              ; Signal similar request found.
```

```
                05  09CF  3080          RSB                                 ; Return error.
                    09D0  3081
                    09D0  3082  100$:    ; Goodie.  No similar request found.
     50   01    D0  09D0  3083          MOVL    #1, R0                      ; Signal no request found.
                05  09D3  3084          RSB                                 ; Return success.
```

```
                                        09D4  3086          .SBTTL  DUTU$BUILD_CANIO_CDRP - Allocate & Initialize a cancel-CDRP
                                        09D4  3087 ;++
                                        09D4  3088 ;
                                        09D4  3089 ; DUTU$BUILD_CANIO_CDRP - Allocate & Initialize a cancel-CDRP
                                        09D4  3090 ;
                                        09D4  3091 ; Functional Description:
                                        09D4  3092 ;
                                        09D4  3093 ;       This routine allocates space for and initializes a cancel-CDRP, the
                                        09D4  3094 ;       IRP/CDRP which represents a cancel operation in progress.  In addition,
                                        09D4  3095 ;       this routine links the new cancel-CDRP to the CDDB queue of active
                                        09D4  3096 ;       cancel-CDRPs.
                                        09D4  3097 ;
                                        09D4  3098 ;       If the cancel-CDRP cannot be allocated, an error status is returned in
                                        09D4  3099 ;       R0.  Otherwise, a success status is returned.
                                        09D4  3100 ;
                                        09D4  3101 ; Inputs:
                                        09D4  3102 ;
                                        09D4  3103 ;       R2      Channel index number on which cancel is to be performed
                                        09D4  3104 ;       R4      Process PCB address
                                        09D4  3105 ;       R5      Device UCB address
                                        09D4  3106 ;
                                        09D4  3107 ; Outputs:
                                        09D4  3108 ;
                                        09D4  3109 ;       R0      SS$_NORMAL cancel-CDRP built without incident
                                        09D4  3110 ;               SS$_INSFMEM insufficient non-paged pool to build cancel-CDRP
                                        09D4  3111 ;       R1      IRP base address of cancel-CDRP (i.e. R1 + IRP$L_FQFL =
                                        09D4  3112 ;               cancel-CDRP base address)
                                        09D4  3113 ;
                                        09D4  3114 ;       All other registers are preserved.
                                        09D4  3115 ;--
                                        09D4  3116
                                        09D4  3117 DUTU$BUILD_CANIO_CDRP:
                                        09D4  3118
                                  3C BB 09D4  3119          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers.
                               51   C4 8F 9A 09D6  3120          MOVZBL  #IRP$C_LENGTH, R1       ; Get size of an IRP/CDRP.
                         00000000'GF 16 09DA  3121          JSB     G^EXE$ALONONPAGED       ; Allocate an IRP/CDRP.
                                  5C 50 E9 09E0  3122          BLBC    R0, 900$                ; Branch if allocation failed.
                                  7E 51 7D 09E3  3123          MOVQ    R1, -(SP)               ; Save allocation size and address.
62   51   00   6E   00 2C 09E6  3124          MOVC5   #0, (SP), #0, R1, (R2)  ; Zero entire allocated region.
                                  3F BA 09EC  3125          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore allocation information
                                        09EE  3126                                  ; and input registers.
                                        09EE  3127
                               08 A1 50 B0 09EE  3128          MOVW    R0, IRP$W_SIZE(R1)      ; Store allocation size.
                                        09F2  3129          ASSUME  IRP$B_RMOD EQ <IRP$B_TYPE + 1>
                               0A A1 0A 9B 09F2  3130          MOVZBW  #DYN$C_IRP, -           ; Store IRP type and mode.
                                        09F6  3131                  IRP$B_TYPE(R1)
                            0C A1   60 A4 D0 09F6  3132          MOVL    PCB$L_PID(R4), -        ; Save canceling process' PID.
                                        09FB  3133                  IRP$L_PID(R1)
                               1C A1 55 D0 09FB  3134          MOVL    R5, IRP$L_UCB(R1)       ; Save UCB of cancel target device.
                               28 A1 52 B0 09FF  3135          MOVW    R2, IRP$W_CHAN(R1)      ; Save cancel channel index.
                                        0A03  3136
                            40 A1   40 A1 9E 0A03  3137          MOVAB   IRP$L_SNDABTQFL(R1), -  ; Initialize the send an ABORT queue.
                                        0A08  3138                  IRP$L_SNDABTQFL(R1)
                            44 A1   40 A1 9E 0A08  3139          MOVAB   IRP$L_SNDABTQFL(R1), -
                                        0A0D  3140                  IRP$L_SNDABTQBL(R1)
                            38 A1   38 A1 9E 0A0D  3141          MOVAB   IRP$L_ABTDQFL(R1), -    ; Initialize the ABORT sent queue.
                                        0A12  3142                  IRP$L_ABTDQFL(R1)
```

```
                3C A1    38 A1    9E   0A12  3143          MOVAB   IRP$L_ABTDQFL(R1), -
                                   0A17  3144                  IRP$L_ABTDQBL(R1)
                                   0A17  3145
                                   0A17  3146          ASSUME  IRP$B_CD_TYPE EQ IRP$W_CDRPSIZE+2
                                   0A17  3147          ASSUME  IRP$B_FIPL    EQ IRP$W_CDRPSIZE+3
        68 A1    0839FFA0 8F      D0   0A17  3148          MOVL    #< <IPL$_SCSA24> -      ; Initialize CDRP size, type and fork
                                   0A1F  3149                    ! <DYN$C_CDRPA16> -     ; IPL fields.
                                   0A1F  3150                    ! <CDRP$C_IOQFL&^xFFFF> >, -
                                   0A1F  3151                  IRP$W_CDRPSIZE(R1)
              0084 C1    00C8 C5    D0   0A1F  3152          MOVL    UCB$L_CDT(R5), -        ; Setup CDT address.
                                   0A26  3153                  IRP$L_CDT(R1)
              0088 C1    56 A5    9E   0A26  3154          MOVAB   UCB$W_RWAITCNT(R5), -   ; Point CDRP to wait counter.
                                   0A2C  3155                  IRP$L_RWCPTR(R1)
                 00A0 C1    02    D0   0A2C  3156          MOVL    #CDRP$M_CANIO, -        ; Mark this as a cancel-CDRP.
                                   0A31  3157                  IRP$L_DUTUFLAGS(R1)
                                   0A31  3158
                 50    00BC C5    D0   0A31  3159          MOVL    UCB$L_CDDB(R5), R0      ; Locate the cancel CDDB.
                 00B4 D0    61    0E   0A36  3160          INSQUE  IRP$L_CANIOQFL(R1), -   ; Insert cancel-CDRP onto CDDB
                                   0A3B  3161                  @CDDB$L_CANCLQBL(R0)    ; queue of cancel-CDRPs.
                                   0A3B  3162
                       50    01    D0   0A3B  3163          MOVL    #SS$_NORMAL, R0         ; Setup success status.
                                   05   0A3E  3164          RSB                             ; Return.
                                   0A3F  3165
                             3C    BA   0A3F  3166 900$:     POPR    #^M<R2,R3,R4,R5>        ; Error exit -- restore registers.
                                   05   0A41  3167          RSB                             ; Return.
```

```
                     0A42  3169          .SBTTL  DUTU$CANCEL_RDTWAIT - Cancel a thread on the RDT wait queue
                     0A42  3170 ;++
                     0A42  3171 ;
                     0A42  3172 ; DUTU$CANCEL_RDTWAIT - Cancel a thread on the RDT wait queue
                     0A42  3173 ;
                     0A42  3174 ; Functional Description:
                     0A42  3175 ;
                     0A42  3176 ;       This is the action routine for SCAN_RSPID_WAIT, the SCS service which
                     0A42  3177 ;       scans the RDT wait queue for entries belonging to a given connection.
                     0A42  3178 ;       After determining that a request presented for processing by this
                     0A42  3179 ;       routine meets the cancel criteria, this routine performs those
                     0A42  3180 ;       operations necessary to cancel the request.
                     0A42  3181 ;
                     0A42  3182 ;       Since a RSPID is a required SCS resource for all MSCP activities,
                     0A42  3183 ;       requests stuck on the RDT wait queue (i.e. requests in a RSPID wait
                     0A42  3184 ;       state), cannot have active MSCP requests outstanding to the server.
                     0A42  3185 ;       Therefore, this routine need only count the need to adjust the wait
                     0A42  3186 ;       count, remove the CDRP from the RDT wait queue, and queue the CDRP
                     0A42  3187 ;       for I/O post processing.
                     0A42  3188 ;
                     0A42  3189 ; Inputs:
                     0A42  3190 ;
                     0A42  3191 ;       R1      cancel-CDRP address
                     0A42  3192 ;       R5      address of a CDRP belonging to the connection on which the
                     0A42  3193 ;               cancel request was made
                     0A42  3194 ;
                     0A42  3195 ; Outputs:
                     0A42  3196 ;
                     0A42  3197 ;       R0, R2, and R5 are destroyed.
                     0A42  3198 ;       All other registers are preserved.
                     0A42  3199 ;--
                     0A42  3200
                     0A42  3201 DUTU$CANCEL_RDTWAIT:
                     0A42  3202
52   55   D0         0A42  3203          MOVL    R5, R2                      ; Copy waiting CDRP address.
55   51   D0         0A45  3204          MOVL    R1, R5                      ; Copy cancel-CDRP address.
                     0A48  3205
                     0A48  3206          IFNOCANCEL cdrp=(R2), then=10$      ; Branch if CDRP shouldn't be canceled.
   44 A5   B6         0A4E  3207          INCW    CDRP$W_DUTUCNTR(R5)         ; Account for wait counter bump due
                     0A51  3208                                              ; to RDT wait state.
50   62   0F         0A51  3209          REMQUE  (R2), R0                    ; Remove CDRP from wait queue and
                     0A54  3210          POST_CDRP status=SS$_CANCEL         ; queue it for I/O post processing.
                     0A61  3211
                05   0A61  3212 10$:     RSB                                 ; Return to RDT wait queue scan.
```

```
                     0A62  3214          .SBTTL  DUTU$CANCEL_RDT - Cancel a thread holding a RSPID
                     0A62  3215 ;++
                     0A62  3216 ;
                     0A62  3217 ; DUTU$CANCEL_RDT - Cancel a thread holding a RSPID
                     0A62  3218 ;
                     0A62  3219 ; Functional Description:
                     0A62  3220 ;
                     0A62  3221 ;       This is the action routine for SCAN_RDT, the SCS service which
                     0A62  3222 ;       Response-id Descriptor Table (RDT) for entries belonging to a given
                     0A62  3223 ;       connection.  After determining that a request presented for processing
                     0A62  3224 ;       by this routine meets the cancel criteria, this routine performs those
                     0A62  3225 ;       operations necessary to cancel the request.
                     0A62  3226 ;
                     0A62  3227 ;       Class driver function threads found in the RDT may or may not
                     0A62  3228 ;       currently have MSCP transactions in progress.  Before canceling are
                     0A62  3229 ;       thread found in the RDT, the presence or absence of an active MSCP
                     0A62  3230 ;       transaction must be determined.  This is done by searching the CDDB
                     0A62  3231 ;       queue of threads with active transactions for the CDRP presented for
                     0A62  3232 ;       processing.
                     0A62  3233 ;
                     0A62  3234 ;       If the CDRP is not found in the CDDB queue, it does not have an active
                     0A62  3235 ;       MSCP transaction.  Therefore, it can be canceled by counting the need
                     0A62  3236 ;       to adjust the wait count, removing the CDRP from the whatever wait
                     0A62  3237 ;       queue it is on, and queuing the CDRP for I/O post processing.
                     0A62  3238 ;
                     0A62  3239 ;       If the CDRP is found in the CDDB queue, it does have an active MSCP
                     0A62  3240 ;       transaction and that transaction must eventually have an MSCP ABORT
                     0A62  3241 ;       command issued against it.  This is done by removing the CDRP from the
                     0A62  3242 ;       CDDB queue and queueing it to the send-abort queue linked to the
                     0A62  3243 ;       cancel-CDRP.  Entries are placed in the send-abort queue in request
                     0A62  3244 ;       sequence number order.  This causes the MSCP ABORT requests to be
                     0A62  3245 ;       issued in the same order as the MSCP commands which they abort were
                     0A62  3246 ;       issued.  This is necessary for proper tape class driver operation.
                     0A62  3247 ;
                     0A62  3248 ; Inputs:
                     0A62  3249 ;
                     0A62  3250 ;       R1      cancel-CDRP address
                     0A62  3251 ;       R5      address of a CDRP belonging to the connection on which the
                     0A62  3252 ;               cancel request was made
                     0A62  3253 ;
                     0A62  3254 ; Outputs:
                     0A62  3255 ;
                     0A62  3256 ;       R0, R2, and R5 are destroyed.
                     0A62  3257 ;       All other registers are preserved.
                     0A62  3258 ;--
                     0A62  3259
                     0A62  3260 DUTU$CANCEL_RDT:
                     0A62  3261
        52   55   D0  0A62  3262          MOVL    R5, R2                          ; Copy RDT CDRP address.
        55   51   D0  0A65  3263          MOVL    R1, R5                          ; Copy cancel-CDRP address.
                     0A68  3264          IFNOCANCEL cdrp=(R2), then=90$          ; Branch if CDRP shouldn't be canceled.
                     0A6E  3265
   51    BC A5   D0  0A6E  3266          MOVL    CDRP$L_UCB(R5), R1              ; Get UCB address.
51  00BC C1   D0  0A72  3267          MOVL    UCB$L_CDDB(R1), R1              ; Get CDDB address.
                     0A77  3268          ASSUME  CDDB$L_CDRPQFL EQ 0
                     0A77  3269          ASSUME  CDRP$L_FQFL    EQ 0
        50   51   D0  0A77  3270          MOVL    R1, R0                          ; Initialize "previous" CDRP address.
```

```
                                0A7A  3271
                50   60   D0    0A7A  3272 10$:    MOVL    CDRP$L_FQFL(R0), R0      ; Link to next CDRP.
                51   50   D1    0A7D  3273         CMPL    R0, R1                   ; Reached end of CDDB queue yet?
                     2C   13    0A80  3274         BEQL    50$                      ; Branch if reached end of queue.
                52   50   D1    0A82  3275         CMPL    R0, R2                   ; Found CDRP to be canceled?
                     F3   12    0A85  3276         BNEQ    10$                      ; Branch if CDRP not found.
                                0A87  3277
                                0A87  3278         ; The CDRP to be canceled has an active MSCP transaction.
                                0A87  3279
                52   62   0F    0A87  3280         REMQUE  (R2), R2                 ; Remove CDRP from CDDB queue.
            40 A2    01   C8    0A8A  3281         BISL    #CDRP$M_CAND, -          ; Flag CDRP as "canceled."
                                0A8E  3282                 CDRP$L_DUTUFLAGS(R2)
            A0 A2    55   D0    0A8E  3283         MOVL    R5, CDRP$L_CANIOCDRP(R2); Set back pointer to cancel-CDRP.
            51   E0 A5   9E    0A92  3284         MOVAB   CDRP$L_SNDABTQFL(R5), R1; Get header for send-abort queue.
                50   51   D0    0A96  3285         MOVL    R1, R0                   ; Initialize "previous" CDRP address.
                                0A99  3286
                50   60   D0    0A99  3287 30$:    MOVL    CDRP$L_FQFL(R0), R0      ; Link to next CDRP.
                51   50   D1    0A9C  3288         CMPL    R0, R1                   ; Reached end of send-abort queue yet?
                     07   13    0A9F  3289         BEQL    35$                      ; Branch if reached end of queue.
        F0 A0    F0 A2   D1    0AA1  3290         CMPL    CDRP$L_SEQNUM(R2), -     ; Is canceled-CDRP seqnum less than
                                0AA6  3291                 CDRP$L_SEQNUM(R0)        ; current send-abort seqnum?
                     F1   1E    0AA6  3292         BGEQU   30$                      ; If not, loop until it is.
                                0AA8  3293
            04 B0    62   0E    0AA8  3294 35$:    INSQUE  (R2), @CDRP$L_FQBL(R0)   ; Insert canceled-CDRP before current
                                0AAC  3295                                          ; send-abort queue entry.
                                0AAC  3296
                     13   11    0AAC  3297         BRB     80$                      ; Join common exit code.
                                0AAE  3298
                                0AAE  3299 50$:    ; The CDRP to be canceled does not have an active MSCP transaction.
                                0AAE  3300
              44 A5    B6    0AAE  3301         INCW    CDRP$W_DUTUCNTR(R5)      ; Account for wait counter bump due
                                0AB1  3302                                          ; to RDT wait state.
        50    62   0F    0AB1  3303         REMQUE  (R2), R0                 ; Remove CDRP from wait queue and
                                0AB4  3304         POST_CDRP status=SS$_CANCEL      ; queue it for I/O post processing.
                                0AC1  3305
        51    55   D0    0AC1  3306 80$:    MOVL    R5, R1                   ; Restore cancel-CDRP address.
                     05    0AC4  3307 90$:    RSB                              ; Return to RDT scan.
```

```
                     0AC5  3309          .SBTTL  DUTU$TEST_CANCEL_DONE - Test for completed cancel operation
                     0AC5  3310 ;++
                     0AC5  3311 ;
                     0AC5  3312 ; DUTU$TEST_CANCEL_DONE - Test for completed cancel operation
                     0AC5  3313 ;
                     0AC5  3314 ; Functional Description:
                     0AC5  3315 ;
                     0AC5  3316 ;       This routine is called by MSCP end message processing whenever an MSCP
                     0AC5  3317 ;       end message is associated with a CDRP which has the CDRP$M_CAND bit
                     0AC5  3318 ;       set in CDRP$L_DUTUFLAGS.  The presence of that flag indicates that a
                     0AC5  3319 ;       cancel-CDRP is waiting for canceled requests to complete.
                     0AC5  3320 ;
                     0AC5  3321 ;       It is assumed that the input CDRP has been removed from whatever queue
                     0AC5  3322 ;       it was on when the MSCP message was received.
                     0AC5  3323 ;
                     0AC5  3324 ;       This routine determines whether or not all canceled requests for the
                     0AC5  3325 ;       cancel-CDRP associated with the input CDRP (which represents one of
                     0AC5  3326 ;       those canceled requests) have completed.  If they all have completed,
                     0AC5  3327 ;       the cancel operation is ended.  Otherwise, no action is taken.
                     0AC5  3328 ;
                     0AC5  3329 ; Inputs:
                     0AC5  3330 ;
                     0AC5  3331 ;       R5      CDRP address (for a canceled request)
                     0AC5  3332 ;
                     0AC5  3333 ; Outputs:
                     0AC5  3334 ;
                     0AC5  3335 ;       All registers are preserved.
                     0AC5  3336 ;--
                     0AC5  3337
                     0AC5  3338 DUTU$TEST_CANCEL_DONE::
                     0AC5  3339
   7E    54    7D    0AC5  3340          MOVQ    R4, -(SP)                       ; Save some registers.
55    A0    A5    D0    0AC8  3341          MOVL    CDRP$L_CANIOCDRP(R5), R5        ; Get cancel-CDRP address.
                     0ACC  3342
54    E0    A5    9E    0ACC  3343          MOVAB   CDRP$L_SNDABTQFL(R5), R4        ; Get send-abort queue header.
   64    54    D1    0AD0  3344          CMPL    R4, (R4)                        ; Is send-abort queue empty?
         0B    12    0AD3  3345          BNEQ    80$                             ; Branch if queue not empty.
54    D8    A5    9E    0AD5  3346          MOVAB   CDRP$L_ABTDQFL(R5), R4          ; Get abort-sent queue header.
   64    54    D1    0AD9  3347          CMPL    R4, (R4)                        ; Is abort-sent queue empty.
         02    12    0ADC  3348          BNEQ    80$                             ; Branch if queue not empty.
                     0ADE  3349
         1C    10    0ADE  3350          BSBB    DUTU$END_CANCEL                 ; Cancel operation is finished.
                     0AE0  3351
   54    8E    7D    0AE0  3352 80$:     MOVQ    (SP)+, R4                       ; Restore saved registers.
               05    0AE3  3353          RSB                                     ; Return.
```

```
                        0AE4  3355          .SBTTL  DUTU$DISCONNECT_CANCEL - Do disconnect cleanup for cancels
                        0AE4  3356 ;++
                        0AE4  3357 ;
                        0AE4  3358 ; DUTU$DISCONNECT_CANCEL - Do disconnect cleanup for cancel requests
                        0AE4  3359 ;
                        0AE4  3360 ; Functional Description:
                        0AE4  3361 ;
                        0AE4  3362 ;       This routine is called to perform those cancel functions required
                        0AE4  3363 ;       whenever a connection breaks.  For each cancel-CDRP on the CDDB cancel
                        0AE4  3364 ;       opeations queue, this routine performs one of two cleanup tasks:
                        0AE4  3365 ;
                        0AE4  3366 ;           o   If the cancel-CDRP has its fork block queued somewhere, it is
                        0AE4  3367 ;               simply removed from the CDDB cancel operations queue.  Some
                        0AE4  3368 ;               other portion of the broken connection cleanup will discover
                        0AE4  3369 ;               the cancel-CDRP in whatever queue its fork block is hung.  The
                        0AE4  3370 ;               cancel-CDRP will then be delivered to DUTU$INSERT_RESTRTQ
                        0AE4  3371 ;               which will complete the needed cancel cleanup.  This cleanup
                        0AE4  3372 ;               is done in this way because this routine cannot know the
                        0AE4  3373 ;               proper connection-dependent cleanup required by the
                        0AE4  3374 ;               cancel-CDRP as a result of its in progress SCS activities.
                        0AE4  3375 ;               N.B. proper operation of this test depends upon SEND_ABORTS
                        0AE4  3376 ;               clearing the cancel-CDRP fork block queue links when it is
                        0AE4  3377 ;               done sending the needed MSCP ABORT commands.
                        0AE4  3378 ;
                        0AE4  3379 ;           o   If the cancel-CDRP fork block is not queue anywhere, the
                        0AE4  3380 ;               cancel-CDRP is dequeued from the CDDB cancel operations queue
                        0AE4  3381 ;               and the cancel operation it represents is ended now.  All
                        0AE4  3382 ;               CDRPs linked to the cancel-CDRP (awaiting completion of cancel
                        0AE4  3383 ;               processing) are immediately queued for I/O post processing.
                        0AE4  3384 ;
                        0AE4  3385 ; Inputs:
                        0AE4  3386 ;
                        0AE4  3387 ;       R3      CDDB address
                        0AE4  3388 ;
                        0AE4  3389 ; Outputs:
                        0AE4  3390 ;
                        0AE4  3391 ;       R0 and R5 are destroyed.
                        0AE4  3392 ;       All other registers are preserved.
                        0AE4  3393 ;--
                        0AE4  3394
                        0AE4  3395 DUTU$DISCONNECT_CANCEL::
                        0AE4  3396
     55   00B0 D3   0F  0AE4  3397 10$:    REMQUE  @CDDB$L_CANCLQFL(R3), R5        ; Get a cancel-CDRP.
                  10   1D  0AE9  3398          BVS     90$                         ; Branch if no cancel-CDRPs.
                        0AEB  3399
                        0AEB  3400          ASSUME  IRP$L_CANIOQFL EQ IRP$L_IOQFL
                        0AEB  3401          ASSUME  IRP$L_IOQFL EQ 0
50   64 A5    60 A5   C9  0AEB  3402          BISL3   IRP$L_FQFL(R5), -            ; Is CDRP fork block queue?
                        0AF1  3403                  IRP$L_FQBL(R5), R0          ; If so, skip further
                  F1   12  0AF1  3404          BNEQ    10$                         ; processing.
                        0AF3  3405
     55   60 A5    9E  0AF3  3406          MOVAB   IRP$L_FQFL(R5), R5          ; Else, cleanup cancel
                  09   10  0AF7  3407          BSBB    DUTU$CLEANUP_CANCEL         ; operation.
                  E9   11  0AF9  3408          BRB     10$                         ; Loop till no more CDRPs.
                        0AFB  3409
                  05        0AFB  3410 90$:    RSB                                 ; Return.
```

```
                     0AFC  3412          .SBTTL  DUTU$END_CANCEL - Finish a cancel operation
                     0AFC  3413          .SBTTL  DUTU$CLEANUP_CANCEL - Cleanup a cancel operation
                     0AFC  3414 ;++
                     0AFC  3415 ;
                     0AFC  3416 ; DUTU$END_CANCEL - Finish a cancel operation
                     0AFC  3417 ; DUTU$CLEANUP_CANCEL - Cleanup a cancel operation
                     0AFC  3418 ;
                     0AFC  3419 ; Functional Description:
                     0AFC  3420 ;
                     0AFC  3421 ;       These routines are called end an in progress cancel operation.  Both
                     0AFC  3422 ;       routines guarantee that all canceled requests not previously posted
                     0AFC  3423 ;       for I/O post processing (or otherwise disposed of) are queued for I/O
                     0AFC  3424 ;       post processing.  Both routines deallocate the cancel-CDRP.
                     0AFC  3425 ;
                     0AFC  3426 ;       DUTU$END_CANCEL removes the cancel-CDRP from the CDDB cancel
                     0AFC  3427 ;       operations queue.  DUTU$CLEANUP_CANCEL does not.
                     0AFC  3428 ;
                     0AFC  3429 ; Inputs:
                     0AFC  3430 ;
                     0AFC  3431 ;       R5      cancel-CDRP address
                     0AFC  3432 ;
                     0AFC  3433 ; Implicit Inputs:
                     0AFC  3434 ;
                     0AFC  3435 ;       CDRP$L_SNDABTQFL(R5)    send-abort queue header
                     0AFC  3436 ;       CDRP$L_ABTDQFL(R5)      abort-sent queue header
                     0AFC  3437 ;       CDRP$L_CANIOQFL(R5)     CDDB cancel operations queue links
                     0AFC  3438 ;
                     0AFC  3439 ; Outputs:
                     0AFC  3440 ;
                     0AFC  3441 ;       All registers preserved.
                     0AFC  3442 ;       CDRP pointed to by R5 is deallocated.
                     0AFC  3443 ;--
                     0AFC  3444
                     0AFC  3445 DUTU$END_CANCEL:
                     0AFC  3446
7E   A0 A5   0F      0AFC  3447          REMQUE  CDRP$L_CANIOQFL(R5), -  ; Remove cancel-CDRP from CDDB cancel
                     0B00  3448                  -(SP)                   ; operations queue.
        8E   D5      0B00  3449          TSTL    (SP)+                   ; Cleanup stack.
                     0B02  3450
                     0B02  3451 DUTU$CLEANUP_CANCEL:
                     0B02  3452
        3F   BB      0B02  3453          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers.
                     0B04  3454
                     0B04  3455          ; First drain any CDRP's that didn't even get there ABORT's sent.
                     0B04  3456
50   E0 B5   0F      0B04  3457 20$:     REMQUE  @CDRP$L_SNDABTQFL(R5), R0 ; Get next canceled IRP/CDRP.
           0F   1D   0B08  3458          BVS     40$                     ; Branch if no more IRPs/CDRPs.
                     0B0A  3459          POST_CDRP status=SS$_CANCEL     ; Insert packet onto IOPOST queue.
        EB   11      0B17  3460          BRB     20$                     ; Loop back until empty.
                     0B19  3461
                     0B19  3462          ; Then drain CDRP's which did not terminate after an ABORT.
                     0B19  3463
50   D8 B5   0F      0B19  3464 40$:     REMQUE  @CDRP$L_ABTDQFL(R5), R0 ; Get next canceled IRP/CDRP.
           0F   1D   0B1D  3465          BVS     60$                     ; Branch if no more IRPs/CDRPs.
                     0B1F  3466          POST_CDRP status=SS$_CANCEL     ; Insert packet onto IOPOST queue.
        EB   11      0B2C  3467          BRB     40$                     ; Loop back until empty.
                     0B2E  3468
```

```
                        0B2E  3469 60$:    ; Deallocate cancel-CDRP.
                        0B2E  3470
                        0B2E  3471          MOVAB   CDRP$L_IOQFL(R5), R0    ; Point to IRP portion.
                        0B32  3472          JSB     G^COM$DRVDEALMEM        ; Deallocate IRP.
                        0B38  3473
                        0B38  3474          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                        0B3A  3475          RSB
```

```
                     0B3B  3477          .SBTTL  DUTU$TEST_CANCEL_CDRP - Should a CDRP be canceled
                     0B3B  3478 ;++
                     0B3B  3479 ;
                     0B3B  3480 ; DUTU$TEST_CANCEL_CDRP - Should a CDRP be canceled
                     0B3B  3481 ;
                     0B3B  3482 ; Functional Description:
                     0B3B  3483 ;
                     0B3B  3484 ;       This routine tests one input CDRP against an input cancel CDRP to
                     0B3B  3485 ;       determine whether or not the former should be canceled.  The CDRP
                     0B3B  3486 ;       should be canceled if and only if the following fields match in both
                     0B3B  3487 ;       CDRPs:
                     0B3B  3488 ;
                     0B3B  3489 ;               CDRP$L_PID              requesting process PID
                     0B3B  3490 ;               CDRP$W_CHAN             requesting process channel
                     0B3B  3491 ;               CDRP$L_UCB              device UCB address
                     0B3B  3492 ;
                     0B3B  3493 ;       NOTE: virtual requests are canceled in the same manner as other
                     0B3B  3494 ;       requests.  This differs from the approach used by other disk drivers.
                     0B3B  3495 ;
                     0B3B  3496 ; Inputs:
                     0B3B  3497 ;
                     0B3B  3498 ;       R2      test CDRP address
                     0B3B  3499 ;       R5      cancel CDRP address
                     0B3B  3500 ;
                     0B3B  3501 ; Outputs:
                     0B3B  3502 ;
                     0B3B  3503 ;       R0      success implies test CDRP should be canceled
                     0B3B  3504 ;               failure implies test CDRP should not be canceled.
                     0B3B  3505 ;--
                     0B3B  3506
                     0B3B  3507 DUTU$TEST_CANCEL_CDRP::
                     0B3B  3508
          50   D4    0B3B  3509          CLRL    R0                                  ; Assume no cancel.
BC A5     BC A2   D1 0B3D  3510          CMPL    CDRP$L_UCB(R2), CDRP$L_UCB(R5)      ; Do devices match?
          10   12   0B42  3511          BNEQ    90$                                 ; Branch if no match.
AC A5     AC A2   D1 0B44  3512          CMPL    CDRP$L_PID(R2), CDRP$L_PID(R5)      ; Do PIDs match?
          09   12   0B49  3513          BNEQ    90$                                 ; Branch if no match.
C8 A5     C8 A2   B1 0B4B  3514          CMPW    CDRP$W_CHAN(R2), CDRP$W_CHAN(R5)    ; Do channels match?
          02   12   0B50  3515          BNEQ    90$                                 ; Branch if no match.
          50   D6   0B52  3516          INCL    R0                                  ; All match, so cancel it.
               05   0B54  3517 90$:     RSB
```

```
                     0B55  3519          .SBTTL  ----- GENERAL SUPPORT ROUTINES -----
                     0B55  3520          .SBTTL  DUTU$RESET_MSCP_MSG - Reset MSCP command packet
                     0B55  3521 ;++
                     0B55  3522 ;
                     0B55  3523 ; DUTU$RESET_MSCP_MSG - Reset MSCP command packet
                     0B55  3524 ;
                     0B55  3525 ; Functional Description:
                     0B55  3526 ;
                     0B55  3527 ;       This routine causes a previously used MSCP command packet (or end
                     0B55  3528 ;       message) to be completely recycled and readied for use in the sending
                     0B55  3529 ;       of another MSCP command.  It is intended for use by those functions
                     0B55  3530 ;       which require more than one MSCP command to properly complete.  Any
                     0B55  3531 ;       failure of the message buffer operation is taken as an indication of a
                     0B55  3532 ;       broken connection and control is transfered to DUTU$KILL_THIS_THREAD.
                     0B55  3533 ;
                     0B55  3534 ;       This routine is the power behind the RESET_MSCP_MSG macro.
                     0B55  3535 ;
                     0B55  3536 ; Inputs:
                     0B55  3537 ;
                     0B55  3538 ;       R3      UCB address
                     0B55  3539 ;       R5      CDRP address
                     0B55  3540 ;
                     0B55  3541 ; Outputs:
                     0B55  3542 ;
                     0B55  3543 ;       R0 & R1 destroyed
                     0B55  3544 ;       R2      reset MSCP command packet address
                     0B55  3545 ;       All other registers preserved.
                     0B55  3546 ;--
                     0B55  3547
                     0B55  3548 DUTU$RESET_MSCP_MSG::
                     0B55  3549
A4 A5 8ED0           0B55  3550          POPL    CDRP$L_IOQBL(R5)                ; Save caller's return address.
                     0B59  3551          RECYCL_RSPID                            ; Recycle the RSPID.
                     0B5F  3552          RECYCL_MSG_BUF                          ; Recycle the MSCP packet.
05 50   E9           0B62  3553          BLBC    R0, 99$                         ; Branch if connection broken.
A4 A5   DD           0B65  3554          PUSHL   CDRP$L_IOQBL(R5)                ; Restore return address.
   09   11           0B68  3555          BRB     DUTU$INIT_MSCP_MSG_UNIT         ; Initialize MSCP packet.
                     0B6A  3556
                     0B6A  3557 99$:     ; RECYCL_MSG_BUF detected a broken connection.
                     0B6A  3558          ; Therefore, kill this execution thread.
                     0B6A  3559
  8E   D5            0B6A  3560          TSTL    (SP)+                           ; Forget return address.
 01F6  31            0B6C  3561          BRW     DUTU$KILL_THIS_THREAD           ; Then kill this thread.
```

```
                        0B6F  3563          .SBTTL  DUTU$INIT_MSCP_MSG_UNIT - Initialize a MSCP message w/ unit number
                        0B6F  3564          .SBTTL  DUTU$INIT_MSCP_MSG - Initialize a MSCP message
                        0B6F  3565 ;++
                        0B6F  3566 ;
                        0B6F  3567 ; DUTU$INIT_MSCP_MSG_UNIT - Initialize a MSCP message w/ unit number
                        0B6F  3568 ; DUTU$INIT_MSCP_MSG - Initialize a MSCP message
                        0B6F  3569 ;
                        0B6F  3570 ; Functional Description:
                        0B6F  3571 ;
                        0B6F  3572 ;       These routines initialize an SCS message buffer for use in transmitting
                        0B6F  3573 ;       a MSCP command packet.  The primary purpose of this initialization is
                        0B6F  3574 ;       to insure that all reserved fields are zero.  In addition, the command
                        0B6F  3575 ;       reference number is loaded with the RSPID.  DUTU$INIT_MSCP_MSG_UNIT
                        0B6F  3576 ;       also loads the unit number from UCB$W_MSCPUNIT.
                        0B6F  3577 ;
                        0B6F  3578 ;       These routines are the power behind the INIT_MSCP_MSG macro.
                        0B6F  3579 ;
                        0B6F  3580 ; Inputs:
                        0B6F  3581 ;
                        0B6F  3582 ;       R3      a UCB address (DUTU$INIT_MSCP_MSG_UNIT only)
                        0B6F  3583 ;       R5      a CDRP address
                        0B6F  3584 ;
                        0B6F  3585 ; Implicit Inputs:
                        0B6F  3586 ;
                        0B6F  3587 ;       CDRP$L_MSG_BUF(R5)      SCS message buffer address
                        0B6F  3588 ;       CDRP$L_RSPID(R5)        SCS RSPID
                        0B6F  3589 ;       UCB$W_MSCPUNIT(R3)      unit number (DUTU$INIT_MSCP_MSG_UNIT only)
                        0B6F  3590 ;
                        0B6F  3591 ; Outputs:
                        0B6F  3592 ;
                        0B6F  3593 ;       R0 & R1 destroyed
                        0B6F  3594 ;       R2      message buffer address
                        0B6F  3595 ;       All other registers are preserved.
                        0B6F  3596 ;
                        0B6F  3597 ; Implicit Outputs:
                        0B6F  3598 ;
                        0B6F  3599 ;       MSCP$K_MXCMDLEN bytes of message buffer are zeroed.
                        0B6F  3600 ;
                        0B6F  3601 ;       MSCP$L_CMD_RED(R2)      <== CDRP$L_RSPID(R5)
                        0B6F  3602 ;
                        0B6F  3603 ;               (DUTU$INIT_MSCP_MSG_UNIT only)
                        0B6F  3604 ;       MSCP$W_UNIT(R2)         <== UCB$W_MSCPUNIT(R3)
                        0B6F  3605 ;
                        0B6F  3606 ; Special notes:
                        0B6F  3607 ;
                        0B6F  3608 ;       These routines are somewhat inefficient and therefore are not suitable
                        0B6F  3609 ;       for code paths requiring speedy execution.
                        0B6F  3610 ;--
                        0B6F  3611
                        0B6F  3612          .ENABLE LSB
                        0B6F  3613
                        0B6F  3614 DUTU$INIT_MSCP_MSG::
              51   D4   0B6F  3615          CLRL    R1                      ; Signal no unit number present.
              05   11   0B71  3616          BRB     10$                     ; Join common code.
                        0B73  3617
                        0B73  3618 DUTU$INIT_MSCP_MSG_UNIT::
51    00D4 C3   3C   0B73  3619          MOVZWL  UCB$W_MSCPUNIT(R3), R1  ; Get unit number.
```

```
                     0B78  3620
52   1C A5   D0      0B78  3621 10$:    MOVL    CDRP$L_MSG_BUF(R5), R2          ; Get message buffer address.
                     0B7C  3622
   50   52   D0      0B7C  3623         MOVL    R2, R0                          ; Copy message buffer address.
                     0B7F  3624         .REPEAT MSCP$K_MXCMDLEN / 8
                     0B7F  3625         CLRQ    (R0)+                           ; Zero entire message buffer.
        80   7C      0B7F  3626         .ENDR
        80   D4      0B87  3627         .IIF    NE MSCP$K_MXCMDLEN & 4, CLRL (R0)+
                     0B89  3628         .IIF    NE MSCP$K_MXCMDLEN & 2, CLRW (R0)+
                     0B89  3629         .IIF    NE MSCP$K_MXCMDLEN & 1, CLRB (R0)+
                     0B89  3630
62   20 A5   D0      0B89  3631         MOVL    CDRP$L_RSPID(R5), -             ; Setup command reference
                     0B8D  3632                 MSCP$L_CMD_REF(R2)              ; number from RSPID.
                     0B8D  3633
04 A2   51   B0      0B8D  3634         MOVW    R1, MSCP$W_UNIT(R2)             ; Setup unit number or zero.
                     0B91  3635
                05   0B91  3636         RSB                                     ; Return to caller.
                     0B92  3637         .DISABLE LSB
```

```
                                        .SBTTL  DUTU$SEND_DRIVER_MSG - Send driver internal message to MSCP server
                                        .SBTTL  DUTU$SEND_MSCP_MSG - Send command message to MSCP server
                                ;++
                                ;
                                ; DUTU$SEND_DRIVER_MSG - Send driver internal message to MSCP server
                                ; DUTU$SEND_MSCP_MSG - Send command message to MSCP server
                                ;
                                ; Functional Description:
                                ;
                                ;       These routines cause the MSCP message packet pointed to by the CDRP
                                ;       whose address is stored in R5 to be transmitted to the MSCP server
                                ;       at the other end of the connection whose PDT address is in R4.  For
                                ;       accounting purposes, the CDRP is queued to the active transfers queue
                                ;       of the CDDB whose address is in UCB$L_CDDB(R3) [DUTU$SEND_DRIVER_MSG
                                ;       uses the permanent CDDB associated with the input CDRP].
                                ;
                                ;       Control is returned to the instruction following the call to this
                                ;       routine when the corresponding MSCP end message has been received.
                                ;
                                ;  Inputs:
                                ;
                                ;       R3 UCB address (DUTU$SEND_MSCP_MSG only)
                                ;       R4 PDT address
                                ;       R5 CDRP address
                                ;       IPL is IPL$_SCS
                                ;
                                ; Outputs:
                                ;
                                ;       R3 - R5 preserved, all other registers destroyed
                                ;       IPL is IPL$_SCS
                                ;--
                                        .ENABLE LSB
                                DUTU$SEND_DRIVER_MSG::
                                        PERMCDRP_TO_CDDB -              ; Locate interesting CDDB.
                                                cdrp=R5, cddb=R1
                        0D   11         BRB     100$                    ; Join common code.
                                DUTU$SEND_MSCP_MSG::
                                        ; The same macro which defines inline versions of SEND_MSCP_MSG
                                        ; is used here to produce the body of this routine.
                                        SEND_MSCP_MSG ROUTINE 100$
CA A5   0080 8F     B3          BITW    #IRP$M_DIAGBUF, CDRP$W_STS(R5)
                    0F   12     BNEQ    30019$
   51   00BC C3     D0          MOVL    UCB$L_CDDB(R3), R1
     04 B1   65     0E          INSQUE  (R5),@CDDB$L_CDRPQBL(R1)
        51   24     D0          MOVL    #MSCP$K_MXCMDLEN,R1
          60 B4     17          JMP     @PDT$L_SNDCNTMSG(R4)
           02CE     30          BSBW    DUTU$DUMP_COMMAND
             EC     11          BRB     30020$
                                        .DISABLE LSB
```

```
                        .SBTTL  DUTU$INTR_ACTION_XFER - Interpret transfer action table
                        .SBTTL  DUTU$INTR_ACTION_N - Interpret non-transfer action table
                ;++
                ;
                ; DUTU$INTR_ACTION_XFER - Interpret transfer action table
                ; DUTU$INTR_ACTION_N - Interpret non-transfer action table
                ;
                ; Functional Description:
                ;
                ;       These routines interpret an action table.  Based upon the comparison
                ;       of a MSCP$W_STATUS value to entries in the action table, the contents
                ;       of R0 (as status code) are set and control is transfered to a
                ;       specified location.
                ;
                ;       An action table is formed an processed using the following sequence of
                ;       macro statements:
                ;
                ;               DO_ACTION {TRANSFER ! NONTRANSFER}
                ;               ACTION_ENTRY mscp_code_1,ss_code_1,destination_1
                ;               ACTION_ENTRY mscp_code_2,ss_code_2,destination_2
                ;               ACTION_ENTRY mscp_code_3,ss_code_3,destination_3
                ;                     .          .          .          .
                ;                     .          .          .          .
                ;                     .          .          .          .
                ;               ACTION_ENTRY mscp_code_n,ss_code_n,destination_n
                ;               ACTION_ENTRY END
                ;
                ;       These routines compare the MSCP end-packet status code found at
                ;       MSCP$W_STATUS(R2) to each mscp_code_x in the action table.  Whenever a
                ;       match is found, ss_code_x is loaded into R0 and control is transfered
                ;       to destination_x.  If the DO_ACTION parameter is NONTRANSFER, the
                ;       DUTU$INTR_ACTION_N entry point is used and the resulting R0 will have
                ;       ss_code_x in the low-order word with the high-order word zero.  If the
                ;       DO_ACTION parameter is TRANSFER, the DUTU$INTR_ACTION_XFER entry point
                ;       is used and the resulting R0 will have ss_code_x in the high-order
                ;       word with the low-order word zero.  This later form is use for
                ;       conveniently forming the final IOSB status field for a transfer
                ;       command, which is of the form:
                ;
                ;               +----------------------+----------------------+
                ;               |   BCNT (low-order)   !        status        |
                ;               +----------------------+----------------------+
                ;               |                      !  BCNT (high-order)   |
                ;               +----------------------+----------------------+
                ;
                ;       The byte count, BCNT, to be returned is loaded into R1.  Then a
                ;       quadword shift is performed moving status into the low-order R0 and
                ;       moving BCNT into the position shown above.
                ;
                ;       Whenever an input MSCP end-packet status cannot be located in the
                ;       action table, control is returned to the first instruction following
                ;       the action table and the contents of R0 and R1 are indeteminant.
                ;
                ; Inputs:
                ;
                ;       R2      MSCP end-packet address
                ;       (SP)    action table base address
```

```
                     0BB7  3745 ;
                     0BB7  3746 ; Implicit Inputs:
                     0BB7  3747 ;
                     0BB7  3748 ;       MSCP$W_STATUS(R2)       MSCP end-packet status
                     0BB7  3749 ;
                     0BB7  3750 ; Outputs:
                     0BB7  3751 ;
                     0BB7  3752 ;       R0      SS$_xxx status value from action table; in low-order R0 for
                     0BB7  3753 ;               DUTU$INTR_ACTION_N, DO_ACTION NONTRANSFER; in high-order R0
                     0BB7  3754 ;               for DUTU$INTR_ACTION_XFER, DO_ACTION TRANSFER
                     0BB7  3755 ;       R1      Corrupted
                     0BB7  3756 ;
                     0BB7  3757 ;       All other registers preserved.
                     0BB7  3758 ;
                     0BB7  3759 ; Implicit Outputs:
                     0BB7  3760 ;
                     0BB7  3761 ;       None.
                     0BB7  3762 ;
                     0BB7  3763 ; Side Effects:
                     0BB7  3764 ;
                     0BB7  3765 ;       Control is transfered to the destination specified in the selected
                     0BB7  3766 ;       action table entry.
                     0BB7  3767 ;--
                     0BB7  3768
                     0BB7  3769 DUTU$INTR_ACTION_XFER::
                     0BB7  3770
     51    10    90  0BB7  3771          MOVB    #16, R1                         ; Set transfer shift count.
           02    11  0BBA  3772          BRB     INTR_ACTION_COMMON              ; Join common code.
                     0BBC  3773
                     0BBC  3774 DUTU$INTR_ACTION_N::
                     0BBC  3775
                 51  94  0BBC  3776          CLRB    R1                              ; Set non-transfer shift count.
                     0BBE  3777
                     0BBE  3778 INTR_ACTION_COMMON:
                     0BBE  3779
           50    8E  05  C3  0BBE  3780          SUBL3   #ATE_ENTRY_LEN, (SP)+, R0       ; Get initial action tbl. addr.
                 7E  51  90  0BC2  3781          MOVB    R1, -(SP)                       ; Save shift count.
                     0BC5  3782
                     0BC5  3783          ASSUME  MSCP$S_ST_MASK LE 7
                     0BC5  3784          ASSUME  MSCP$V_ST_MASK EQ 0
     51    0A A2    E0  8F  8B  0BC5  3785          BICB3   #^cMSCP$M_ST_MASK, -            ; Get MSCP end-packet status
                     0BCB  3786                  MSCP$W_STATUS(R2), R1           ; value.
                     0BCB  3787
                 50  05  C0  0BCB  3788 10$:     ADDL    #ATE_ENTRY_LEN, R0              ; Move to next action entry.
                     60  B5  0BCE  3789          TSTW    ATE_OFFSET(R0)                  ; Is this end of action table?
                     16  13  0BD0  3790          BEQL    90$                             ; Branch if end of table.
               51    02 A0  91  0BD2  3791          CMPB    ATE_MSCPCODE(R0), R1            ; Compare MSCP status.
                     F3  12  0BD6  3792          BNEQ    10$                             ; Loop if wrong MSCP status.
                     0BD8  3793
                 51  60  32  0BD8  3794          CVTWL   ATE_OFFSET(R0), R1              ; Get destination displacement
                     0BDB  3795          ASSUME  ATE_OFFSET EQ 0                 ; from the action table.
                 51  50  C0  0BDB  3796          ADDL    R0, R1                          ; Convert to absolute address.
           50      03 A0  3C  0BDE  3797          MOVZWL  ATE_SSCODE(R0), R0              ; Get SS$_xxx code from table.
     50    50    8E  9C  0BE2  3798          ROTL    (SP)+, R0, R0                   ; Shift if by shift count.
                     61  17  0BE6  3799          JMP     (R1)                            ; Go to specified destination.
                     0BE8  3800
                     0BE8  3801 90$:     ; MSCP status not found in action table.
```

```
                8E   94  0BE8  3802          CLRB    (SP)+                   ; Pop shift count from stack.
            02  A0   17  0BEA  3803          JMP     2(R0)                   ; Continue after end of action
                      0BED  3804                                            ; table.
```

```
                     0BED  3806          .SBTTL  DUTU$RESTORE_CREDIT - Restore allocated message credit
                     0BED  3807 ;++
                     0BED  3808 ;
                     0BED  3809 ; DUTU$RESTORE_CREDIT - Restore allocated message credit
                     0BED  3810 ;
                     0BED  3811 ; Functional Description:
                     0BED  3812 ;
                     0BED  3813 ;       An SCS message buffer has been allocated, but its use is no longer
                     0BED  3814 ;       needed.  However, the buffer cannot simple be deallocated, because to
                     0BED  3815 ;       do so would result in permanently loosing a send credit to the MSCP
                     0BED  3816 ;       server.  Therefore, a NOP MSCP command must be sent to the MSCP server.
                     0BED  3817 ;       This will cause it to restore the allocated (and now used) send
                     0BED  3818 ;       credit.
                     0BED  3819 ;
                     0BED  3820 ;       To accomplish its goals, this routine plays fast an loose with the
                     0BED  3821 ;       port driver.  A send message request is queued, but before the
                     0BED  3822 ;       response can be received, the RSPID for the request is deallocated.
                     0BED  3823 ;       This is guaranteed by this routine no breaking IPL$_SCS synchronization
                     0BED  3824 ;       from the time the request is queued until the time the RSPID is
                     0BED  3825 ;       deallocated and the CDRP is dequeued from the CDDB active requests
                     0BED  3826 ;       queue (work queue).  Because the RSPID is no longer allocated when the
                     0BED  3827 ;       response message for the NOP command is received, the class driver IDR
                     0BED  3828 ;       routine will discard the response message, which is exactly the
                     0BED  3829 ;       desired effect.
                     0BED  3830 ;
                     0BED  3831 ; Inputs:
                     0BED  3832 ;
                     0BED  3833 ;       R3      UCB address
                     0BED  3834 ;       R4      PDT address
                     0BED  3835 ;       R5      CDRP address
                     0BED  3836 ;
                     0BED  3837 ; Implicit Inputs:
                     0BED  3838 ;
                     0BED  3839 ;       CDRP$L_MSG_BUF(R5)      message buffer address
                     0BED  3840 ;       CDRP$L_RSPID(R5)        RSPID
                     0BED  3841 ;       UCB$W_MSCPUNIT(R3)      unit number
                     0BED  3842 ;
                     0BED  3843 ; Outputs:
                     0BED  3844 ;
                     0BED  3845 ;       All registers preserved.
                     0BED  3846 ;
                     0BED  3847 ; Implicit Outputs:
                     0BED  3848 ;
                     0BED  3849 ;       CDRP$L_MSG_BUF(R5)      message buffer address - deallocated
                     0BED  3850 ;       CDRP$L_RSPID(R5)        RSPID - deallocated
                     0BED  3851 ;--
                     0BED  3852
                     0BED  3853 DUTU$RESTORE_CREDIT::
                     0BED  3854
          0F   BB    0BED  3855          PUSHR   #^M<R0,R1,R2,R3>                ; Save registers.
        7E   54  7D  0BEF  3856          MOVQ    R4, -(SP)                       ; Save PDT and CDRP.
                     0BF2  3857          INIT_MSCP_MSG ucb=(R3)                  ; Ready packet to send NOP.
  08 A2   03  90     0BF5  3858          MOVB    #MSCP$K_OP_GTUNT, -             ; Use Get Unit Status function
                     0BF9  3859                  MSCP$B_OPCODE(R2)               ; as NOP.
    00'AF   9F       0BF9  3860          PUSHAB  B^10$                           ; Set inline caller's caller.
                     0BFC  3861          SEND_MSCP_MSG                           ; Send NOP command.
            05       0BFF  3862          RSB                                     ; If ever get here, kill this
```

```
                     0C00  3863                                          ; fork thread.
                     0C00  3864
                     0C00  3865 10$:     ; Control is returned synchronously returned here as soon as the NOP
                     0C00  3866          ; message is sent.  The fork thread is evaporated by deallocating its
                     0C00  3867          ; RSPID and removing the CDRP from the class driver work queue.
54   8E   7D        0C00  3868          MOVQ    (SP)+, R4                 ; Restore PDT and CDRP.
                     0C03  3869          DEALLOC_RSPID                     ; Quick, deallocate RSPID.
55   65   0F        0C09  3870          REMQUE  (R5), R5                  ; Remove CDRP from work queue.
     0F   BA        0C0C  3871          POPR    #^M<R0,R1,R2,R3>          ; Restore other registers.
          05        0C0E  3872          RSB                               ; Return to caller.
```

```
0C0F  3874          .SBTTL  DUTU$INSERT_RESTARTQ - Insert outstanding CDRP in restart queue
0C0F  3875  ;++
0C0F  3876  ;
0C0F  3877  ; DUTU$INSERT_RESTARTQ - Insert outstanding CDRP in restart queue
0C0F  3878  ;
0C0F  3879  ; Functional Description:
0C0F  3880  ;
0C0F  3881  ;       This routine is presented with outstanding I/O requests (represented
0C0F  3882  ;       by CDRPs) whenever a connection-failure cleanup is required.  For each
0C0F  3883  ;       request, this routine determines the appropriate cleanup action and
0C0F  3884  ;       performs that action.
0C0F  3885  ;
0C0F  3886  ;       In all cases the RSPID and message buffer are deallocated.  N.B.
0C0F  3887  ;       mapping resources are not deallocated.  This action is postponed until
0C0F  3888  ;       after the connection is formally broken, to prevent an "insane" server
0C0F  3889  ;       from incorrectly overwriting memory due to reallocation of mapping
0C0F  3890  ;       resources.
0C0F  3891  ;
0C0F  3892  ;       The various cleanup processing cases are as follows:
0C0F  3893  ;
0C0F  3894  ;           1.  Ordinary IRP/CDRP      - insert the CDRP in the restart queue
0C0F  3895  ;                                        in sequence number order.
0C0F  3896  ;
0C0F  3897  ;           2.  CDDB Permanent         - no action other than deallocating SCS
0C0F  3898  ;               IRP/CDRP                 resources
0C0F  3899  ;
0C0F  3900  ;           3.  HIRT Permanent         - find CDRP which incurrect the
0C0F  3901  ;               IRP/CDRP                 replacement request, process it with a
0C0F  3902  ;                                        recursive call to this routine, copy
0C0F  3903  ;                                        HIRT permanent IRP/CDRP mapping
0C0F  3904  ;                                        resource information to a CDDB
0C0F  3905  ;                                        permanent CDRP, and unlock the HIRT.
0C0F  3906  ;
0C0F  3907  ;           4.  Cancel Operation       - complete the cancel operation, posting
0C0F  3908  ;               IRP/CDRP                 all effected IRP/CDRPs because the
0C0F  3909  ;                                        connection failure has done what the
0C0F  3910  ;                                        ABORT command had (so far) failed to do.
0C0F  3911  ;
0C0F  3912  ;           5.  Mount Verification     - Same as 1.  However, after the mapping
0C0F  3913  ;               IRP/CDRP                 resources are deallocated, these
0C0F  3914  ;                                        IRP/CDRPs will be queued for post
0C0F  3915  ;                                        processing with a status SS$_MEDOFL.
0C0F  3916  ;
0C0F  3917  ; Inputs:
0C0F  3918  ;
0C0F  3919  ;       R3      CDDB address
0C0F  3920  ;       R4      PDT address
0C0F  3921  ;       R5      CDRP address
0C0F  3922  ;
0C0F  3923  ; Implicit inputs:
0C0F  3924  ;
0C0F  3925  ;       T.B.S.
0C0F  3926  ;
0C0F  3927  ; Outputs:
0C0F  3928  ;
0C0F  3929  ;       R0 through R2 and R5 are destroyed.
0C0F  3930  ;       All other registers are preserved.
```

```
                                0C0F  3931 ;
                                0C0F  3932 ; Implicit outputs:
                                0C0F  3933 ;
                                0C0F  3934 ;       See functional description.
                                0C0F  3935 ;--
                                0C0F  3936
                                0C0F  3937 DUTU$INSERT_RESTARTQ::
                                0C0F  3938
      0A A5    39    91        0C0F  3939          CMPB    #DYN$C_CDRP, -          ; Guarantee that a CDRP is being
                                0C13  3940                  CDRP$B_CD_TYPE(R5)      ; processed.
               33    12        0C13  3941          BNEQ    999$                    ; If not, bug check.
                                0C15  3942
            00BB    30        0C15  3943          BSBW    DUTU$DEALLOC_RSPID_MSG  ; Always deallocate RSPID & msg. buf.
                                0C18  3944
   24 40 A5    03    E0        0C18  3945          BBS     #CDRP$V_PERM, -         ; Is this a permanent CDRP?
                                0C1D  3946                  CDRP$L_DUTUFLAGS(R5), - ; Branch if permanent CDRP.
                                0C1D  3947                  200$
   20 40 A5    04    E0        0C1D  3948          BBS     #CDRP$V_HIRT, -         ; Is this a HIRT permanent CDRP?
                                0C22  3949                  CDRP$L_DUTUFLAGS(R5), - ; Branch if HIRT permanent CDRP.
                                0C22  3950                  300$
   1E 40 A5    01    E0        0C22  3951          BBS     #CDRP$V_CANIO, -        ; Is this a CancelIO CDRP?
                                0C27  3952                  CDRP$L_DUTUFLAGS(R5), - ; Branch if CancelIO CDRP.
                                0C27  3953                  400$
                                0C27  3954
      50   3C A3    9E        0C27  3955          MOVAB   CDDB$L_RSTRTQFL(R3), R0 ; Get restart queue list head.
         51    50    D0        0C2B  3956          MOVL    R0, R1                  ; Copy that address.
                                0C2E  3957
         51    61    D0        0C2E  3958 10$:     MOVL    (R1), R1                ; Get next restart CDRP.
         50    51    D1        0C31  3959          CMPL    R1, R0                  ; Returned to the listhead yet?
               07    13        0C34  3960          BEQL    40$                     ; Branch if returned to listhead.
   F0 A1    F0 A5    D1        0C36  3961          CMPL    CDRP$L_SEQNUM(R5), -    ; Is seqnum of CDRP of interest less
                                0C3B  3962                  CDRP$L_SEQNUM(R1)       ; than seqnum of current list entry?
               F1    1E        0C3B  3963          BGEQU   10$                     ; If not, go back to try next list
                                0C3D  3964                                          ; entry.  Else fall thru and insert
                                0C3D  3965                                          ; CDRP of interest before the
                                0C3D  3966                                          ; current entry.
      04 B1    65    0E        0C3D  3967 40$:     INSQUE  (R5),@CDRP$L_FQBL(R1)   ; Insert before current entry.
                                0C41  3968
                     05        0C41  3969 200$:    RSB                             ; Return to caller.
                                0C42  3970
                                0C42  3971 300$:    .WEAK   DU$RSTRTQ_HIRT_CDRP     ; This is not present in tape class
                                0C42  3972                                          ; drivers.
            F3BB'    31        0C42  3973          BRW     DU$RSTRTQ_HIRT_CDRP     ; Process HIRT permanent CDRP and
                                0C45  3974                                          ; return to caller.
                                0C45  3975
            FEBA     31        0C45  3976 400$:    BRW     DUTU$CLEANUP_CANCEL     ; For CancelIO, cleanup cancel and
                                0C48  3977                                          ; return to caller.
                                0C48  3978
                                0C48  3979 999$:    BUG_CHECK MSCPCLASS, FATAL      ; Attempt to insert non-CDRP in the
                                0C4C  3980                                          ; CDDB restart queue.
```

```
OC4C  3982          .SBTTL  DUTU$RECONN_LOOKUP - Reconnection SCS Lookup Action Routine
OC4C  3983 ;++
OC4C  3984 ;
OC4C  3985 ; DUTU$RECONN_LOOKUP - Reconnection SCS Lookup Action Routine
OC4C  3986 ;
OC4C  3987 ; Functional Description:
OC4C  3988 ;
OC4C  3989 ;       During broken connection cleanup, this is the SCS action routine for
OC4C  3990 ;       both SCAN_RSPID_WAIT and SCAN_RDT.  SCAN_RSPID_WAIT is the SCS service
OC4C  3991 ;       which scans the RDT wait queue for entries belonging to a given
OC4C  3992 ;       connection.  SCAN_RDT is the SCS service which scans the Response-id
OC4C  3993 ;       Descriptor Table (RDT) for entries belonging to a given connection.
OC4C  3994 ;       For connection cleanup, the operations performed in both cases are
OC4C  3995 ;       very similar.  For those cases where, some cleanup operation is not
OC4C  3996 ;       appropriate for one or the other situtation, which type of scan is in
OC4C  3997 ;       progress is determined from R1 (as setup by the caller of the scan SCS
OC4C  3998 ;       service).
OC4C  3999 ;
OC4C  4000 ;       CDRPs which have already been canceled are ignored by this routine.
OC4C  4001 ;       They will be properly processed when the request which canceled them
OC4C  4002 ;       is cleaned up.  In all other cases, the CDRP is removed from whatever
OC4C  4003 ;       wait queue it is on.  All CDRPs presented to this routine are assumed
OC4C  4004 ;       to be waiting for resources.  For connection permanent CDRPs, no other
OC4C  4005 ;       processing is required.  In particular, connection permanent CDRPs are
OC4C  4006 ;       never placed on the restart queue.  This is inappropriate because the
OC4C  4007 ;       need for whatever function the connection permanent CDRP was performing
OC4C  4008 ;       disappeared with the broken connection.  Also, since all connection
OC4C  4009 ;       permanent CDRPs are easily located, they need not be queued anywhere
OC4C  4010 ;       to allow easy location when the time comes to release their resources.
OC4C  4011 ;       Connection permanent CDRPs never point to a wait reasons counter, so
OC4C  4012 ;       that processing is not necessary either.
OC4C  4013 ;
OC4C  4014 ;       All other CDRPs are tested for a wait reasons counter pointer.  If one
OC4C  4015 ;       is present, the wait reasons counter is decremented.  CDRPs on the
OC4C  4016 ;       RSPID-wait queue, are obviously waiting for resources.  Otherwise, the
OC4C  4017 ;       fact that all possibly active requests have been cleaned up before
OC4C  4018 ;       this routine is called guarantees that CDRPs presented to this routine
OC4C  4019 ;       are waiting for resources.  Note: this routine helps perpetuate that
OC4C  4020 ;       guarantee by exiting via DUTU$DRAIN_CDDB_CDRPQ, which cleans up all
OC4C  4021 ;       active requests.  Therefore, the cleanup of the CDRP presented to this
OC4C  4022 ;       routine will reduce the number for reasons activity is waiting and the
OC4C  4023 ;       wait reasons counter should be (and is) reduced to reflect this.
OC4C  4024 ;
OC4C  4025 ;       If appropriate, the CDRP is inserted into the restart queue.  Finally,
OC4C  4026 ;       DUTU$DRAIN_CDDB_CDRPQ is called to cleanup and requests made active by
OC4C  4027 ;       the resources released during the cleanup of this CDRP.
OC4C  4028 ;
OC4C  4029 ; Inputs:
OC4C  4030 ;
OC4C  4031 ;       R1      LBC ==> SCAN_RSPID_WAIT
OC4C  4032 ;               LBS ==> SCAN_RDT
OC4C  4033 ;       R3      CDT address
OC4C  4034 ;       R4      PDT address
OC4C  4035 ;       R5      CDRP address
OC4C  4036 ;
OC4C  4037 ; Implicit Inputs:
OC4C  4038 ;
```

```
                          OC4C  4039 ;       CDT$L_AUXSTRUC(R3)      CDDB address
                          OC4C  4040 ;
                          OC4C  4041 ; Outputs:
                          OC4C  4042 ;
                          OC4C  4043 ;       R0, R2, and R5 are destroyed.
                          OC4C  4044 ;       All other registers are preserved.
                          OC4C  4045 ;
                          OC4C  4046 ; Side Effects:
                          OC4C  4047 ;
                          OC4C  4048 ;       The CDRP presented to this routine is readied for retry (or reuse)
                          OC4C  4049 ;       after the a new connection to the remote MSCP server is formed.
                          OC4C  4050 ;--
                          OC4C  4051
                          OC4C  4052 DUTU$RECONN_LOOKUP::
                          OC4C  4053
                          OC4C  4054          ASSUME  CDRP$V_CAND EQ 0                ; Branch if this is a
        1A 40 A5    E8    OC4C  4055          BLBS    CDRP$L_DUTUFLAGS(R5), 50$       ; canceled CDRP.
          55    65    0F  OC50  4056          REMQUE  (R5), R5                        ; Remove CDRP from queue.
12 40 A5    03    E0    OC53  4057          BBS     #CDRP$V_PERM, -                 ; Branch if this is a
                          OC58  4058                  CDRP$L_DUTUFLAGS(R5), 50$       ; permanent CDRP.
   50   28 A5    D0    OC58  4059          MOVL    CDRP$L_RWCPTR(R5), R0           ; Get wait counter pointer.
             02    13    OC5C  4060          BEQL    10$                             ; Branch if no pointer present.
             60    D7    OC5E  4061          DECL    (R0)                            ; Else, decrement wait counter.
             0A    BB    OC60  4062 10$:     PUSHR   #^M<R1,R3>                      ; Save registers.
   53   5C A3    D0    OC62  4063          MOVL    CDT$L_AUXSTRUC(R3), R3          ; Get CDDB address.
             A7    10    OC66  4064          BSBB    DUTU$INSERT_RESTARTQ            ; Setup CDRP restart.
             0A    BA    OC68  4065          POPR    #^M<R1,R3>                      ; Restore registers.
          12 51    E9    OC6A  4066 50$:     BLBC    R1, RECONN_EXIT                 ; If SCAN_RSPID_WAIT, exit now.
                          OC6D  4067 ;------ BRB     DUTU$DRAIN_CDDB_CDRPQ           ; Else, empty active CDRPs queue
                          OC6D  4068                                                  ; and return to caller.
```

```
                        0C6D  4070          .SBTTL  DUTU$DRAIN_CDDB_CDRPQ - Drain Active CDRPs Queue
                        0C6D  4071 ;++
                        0C6D  4072 ;
                        0C6D  4073 ; DUTU$DRAIN_CDDB_CDRPQ - Drain Active CDRPs Queue
                        0C6D  4074 ;
                        0C6D  4075 ; Functional Description:
                        0C6D  4076 ;
                        0C6D  4077 ;       All CDRPs found on the active CDRPs queue, CDDB$L_CDRPQFL, are
                        0C6D  4078 ;       processed and placed on the restart queue, CDDB$L_RSTRTQFL, as
                        0C6D  4079 ;       appropriate.
                        0C6D  4080 ;
                        0C6D  4081 ; Inputs:
                        0C6D  4082 ;
                        0C6D  4083 ;       R3      CDT address
                        0C6D  4084 ;
                        0C6D  4085 ; Implicit Inputs:
                        0C6D  4086 ;
                        0C6D  4087 ;       CDT$L_AUXSTRUC(R3)      CDDB address
                        0C6D  4088 ;
                        0C6D  4089 ; Outputs:
                        0C6D  4090 ;
                        0C6D  4091 ;       All registers preserved.
                        0C6D  4092 ;
                        0C6D  4093 ; Side Effects:
                        0C6D  4094 ;
                        0C6D  4095 ;       Any CDRP found on the active queue is processed by DUTU$INSERT_RESTARTQ.
                        0C6D  4096 ;--
                        0C6D  4097
                        0C6D  4098 DUTU$DRAIN_CDDB_CDRPQ::
                        0C6D  4099
          3F    BB      0C6D  4100          PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Save registers.
53   5C   A3    D0      0C6F  4101          MOVL    CDT$L_AUXSTRUC(R3), R3          ; Get CDDB address.
                        0C73  4102
55   00   B3    0F      0C73  4103 10$:     REMQUE  @CDDB$L_CDRPQFL(R3), R5         ; Get an active CDRP.
               04    1D  0C77  4104          BVS     20$                             ; Branch if no active CDRPs.
               94    10  0C79  4105          BSBB    DUTU$INSERT_RESTARTQ            ; Insert active CDRP in the
               F6    11  0C7B  4106          BRB     10$                             ; restart queue and loop.
                        0C7D  4107
               3F    BA  0C7D  4108 20$:     POPR    #^M<R0,R1,R2,R3,R4,R5>          ; Restore registers.
                        0C7F  4109
                        0C7F  4110 RECONN_EXIT:
                    05  0C7F  4111          RSB                                     ; Return.
```

```
                                0C80  4113          .SBTTL  DUTU$TERMINATE_PENDING - Fail pending requests with SS$_VOLINV
                                0C80  4114 ;++
                                0C80  4115 ;
                                0C80  4116 ; DUTU$TERMINATE_PENDING - Fail pending requests with SS$_VOLINV
                                0C80  4117 ;
                                0C80  4118 ; Functional Description:
                                0C80  4119 ;
                                0C80  4120 ;       This routine locates at queues for I/O post processing all I/O
                                0C80  4121 ;       pending requests on the input UCB.  All requests are completed
                                0C80  4122 ;       with a SS$_VOLINV, "volume is not software enabled" status.
                                0C80  4123 ;
                                0C80  4124 ;       The active requests queue is not scanned by this routine.  Either
                                0C80  4125 ;       there should be not requests in this queue (i.e. there is no good
                                0C80  4126 ;       connection for the requests to be processed on), or the active
                                0C80  4127 ;       requests will eventually be finished, successfully or otherwise by the
                                0C80  4128 ;       MSCP server.
                                0C80  4129 ;
                                0C80  4130 ; Inputs:
                                0C80  4131 ;
                                0C80  4132 ;       R5      UCB address
                                0C80  4133 ;
                                0C80  4134 ; Implicit inputs:
                                0C80  4135 ;
                                0C80  4136 ;       UCB$L_CDDB(R5)          CDDB address
                                0C80  4137 ;       UCB$L_IOQFL(R5)         header for pending I/O request queue
                                0C80  4138 ;       CDDB$L_RSTRTQFL(cddb)   header for queue of I/O requests awaiting
                                0C80  4139 ;                               SCS connection reestablishment
                                0C80  4140 ;
                                0C80  4141 ; Outputs:
                                0C80  4142 ;
                                0C80  4143 ;       R0 through R2 are destroyed.
                                0C80  4144 ;       All other registers are preserved.
                                0C80  4145 ;
                                0C80  4146 ; Implicit outputs:
                                0C80  4147 ;
                                0C80  4148 ;       None.
                                0C80  4149 ;--
                                0C80  4150
                                0C80  4151 DUTU$TERMINATE_PENDING::
                                0C80  4152
 51    00BC C5    3C    C1      0C80  4153          ADDL3   #CDDB$L_RSTRTQFL, -          ; Get address of restart Qhead.
                                0C86  4154                  UCB$L_CDDB(R5), R1
                                0C86  4155          ASSUME  CDRP$L_FQFL EQ 0
          50    51    D0      0C86  4156          MOVL    R1, R0                        ; Init "previous" CDRP pointer.
                                0C89  4157
          50    60    D0      0C89  4158 10$:     MOVL    CDRP$L_FQFL(R0), R0           ; Link to next CDRP.
          51    50    D1      0C8C  4159 11$:     CMPL    R0, R1                        ; All restart CDRPs tested?
                1D    13      0C8F  4160          BEQL    50$                           ; Branch if all CDRPs done.
       55    BC    A0    D1    0C91  4161          CMPL    CDRP$L_UCB(R0), R5            ; Is this CDRP for right UCB?
                F2    12      0C95  4162          BNEQ    10$                           ; Branch if not right UCB.
                60    DD      0C97  4163          PUSHL   CDRP$L_FQFL(R0)               ; Right UCB, save next CDRP
          50    60    0F      0C99  4164          REMQUE  (R0), R0                      ; address and dequeue this CDRP.
                                0C9C  4165          POST_CDRP status=SS$_VOLINV           ; Then, post this CDRP.
                50  8ED0      0CA9  4166          POPL    R0                            ; Restore next CDRP address.
                DE    11      0CAC  4167          BRB     11$                           ; Loop till no more CDRPs.
                                0CAE  4168
       50    4C    B5    0F    0CAE  4169 50$:     REMQUE  @UCB$L_IOQFL(R5), R0          ; Get next pending IRP.
```

```
                13   1D  0CB2  4170          BVS     80$                             ; Branch if no more IRPs.
                     0CB4  4171          POST_IRP status=SS$_VOLINV                  ; Post IRP.
                E7   11  0CC5  4172          BRB     50$                             ; Loop till no more IRPs.
                     0CC7  4173
                     05  0CC7  4174 80$:     RSB                                     ; Return.
```

```
                     0CC8  4176          .SBTTL  DUTU$DEALLOC_ALL - Deallocate all SCS resources
                     0CC8  4177          .SBTTL  DUTU$DEALLOC_RSPID_MSG - Deallocate SCS RSPID and msg. buf.
                     0CC8  4178 ;++
                     0CC8  4179 ;
                     0CC8  4180 ; DUTU$DEALLOC_ALL - Deallocate all SCS resources
                     0CC8  4181 ; DUTU$DEALLOC_RSPID_MSG - Deallocate SCS RSPID and message buffer
                     0CC8  4182 ;
                     0CC8  4183 ; Functional Description:
                     0CC8  4184 ;
                     0CC8  4185 ;       These routines deallocate various combinations of SCS resources.
                     0CC8  4186 ;
                     0CC8  4187 ;       These routines are the power behind the DEALLOCATE macro.
                     0CC8  4188 ;
                     0CC8  4189 ; Inputs:
                     0CC8  4190 ;
                     0CC8  4191 ;       R4      PDT address
                     0CC8  4192 ;       R5      CDRP address
                     0CC8  4193 ;
                     0CC8  4194 ; Outputs:
                     0CC8  4195 ;
                     0CC8  4196 ;       R0 - R2 are destroyed.
                     0CC8  4197 ;       All other registers are preserved.
                     0CC8  4198 ;--
                     0CC8  4199
                     0CC8  4200 DUTU$DEALLOC_ALL::
                     0CC8  4201
2C A5   D5           0CC8  4202          TSTL    CDRP$L_LBUFH_AD(R5)             ; Are map resources allocated?
   06   13           0CCB  4203          BEQL    10$                             ; Branch if not allocated.
                     0CCD  4204          UNMAP                                   ; Else, deallocate them.
2C A5   D4           0CD0  4205          CLRL    CDRP$L_LBUFH_AD(R5)             ; Signal no map stuff allocated.
                     0CD3  4206 10$:
                     0CD3  4207
                     0CD3  4208 DUTU$DEALLOC_RSPID_MSG::
                     0CD3  4209
1C A5   D5           0CD3  4210          TSTL    CDRP$L_MSG_BUF(R5)              ; Is msg. buffer allocated?
   03   13           0CD6  4211          BEQL    20$                             ; Branch if not allocated.
                     0CD8  4212          DEALLOC_MSG_BUF                         ; Else, deallocate it.
                     0CDB  4213 20$:
                     0CDB  4214
20 A5   D5           0CDB  4215          TSTL    CDRP$L_RSPID(R5)                ; Is RSPID allocated?
   06   13           0CDE  4216          BEQL    30$                             ; Branch if not allocated.
                     0CE0  4217          DEALLOC_RSPID                           ; Else, deallocate it.
                     0CE6  4218 30$:
                     0CE6  4219
        05           0CE6  4220          RSB                                     ; Return.
```

```
                                0CE7  4222          .SBTTL  DUTU$POST_CDRP - Queue CDRP for post processing
                                0CE7  4223 ;++
                                0CE7  4224 ;
                                0CE7  4225 ; DUTU$POST_CDRP - Queue CDRP for post processing
                                0CE7  4226 ;
                                0CE7  4227 ; Functional Description:
                                0CE7  4228 ;
                                0CE7  4229 ;       The input CDRP is queued to I/O post processing with the input final
                                0CE7  4230 ;       I/O status.  All SCS resources held by the CDRP are appropriately
                                0CE7  4231 ;       released.
                                0CE7  4232 ;
                                0CE7  4233 ; Inputs:
                                0CE7  4234 ;
                                0CE7  4235 ;       R0      CDRP address
                                0CE7  4236 ;       R1      final I/O status
                                0CE7  4237 ;
                                0CE7  4238 ; Outputs:
                                0CE7  4239 ;
                                0CE7  4240 ;       R0 & R1 are destroyed.
                                0CE7  4241 ;       all other registers are preserved.
                                0CE7  4242 ;--
                                0CE7  4243
                                0CE7  4244 DUTU$POST_CDRP::
                                0CE7  4245
  0A A0   39   91               0CE7  4246          CMPB    #DYN$C_CDRP, -                  ; Guarantee that a CDRP is being
                                0CEB  4247                  CDRP$B_CD_TYPE(R0)              ; processed.
          74   12               0CEB  4248          BNEQ    999$                            ; If not, bug check.
                                0CED  4249
          3C   BB               0CED  4250          PUSHR   #^M<R2,R3,R4,R5>                ; Save registers.
     55   50   D0               0CEF  4251          MOVL    R0, R5                          ; Move CDRP address.
  50    BC A5   D0               0CF2  4252          MOVL    CDRP$L_UCB(R5), R0              ; Get UCB address.
54   0084 C0   D0               0CF6  4253          MOVL    UCB$L_PDT(R0), R4               ; Get PDT address.
                                0CFB  4254
    D8 A5   51   D0             0CFB  4255          MOVL    R1, CDRP$L_IOST1(R5)            ; Set final I/O status.
        DC A5   D4              0CFF  4256          CLRL    CDRP$L_IOST2(R5)                ; Clear second status longword.
                                0D02  4257
        1C A5   D5              0D02  4258          TSTL    CDRP$L_MSG_BUF(R5)              ; Is message buf. allocated?
          03    13              0D05  4259          BEQL    10$                             ; Branch if not allocated.
         FEE3   30              0D07  4260          BSBW    DUTU$RESTORE_CREDIT             ; Else, restore send credit.
                                0D0A  4261
          BC    10              0D0A  4262 10$:     BSBB    DUTU$DEALLOC_ALL                ; Insure that all SCS
                                0D0C  4263                                                  ; resources are deallocated.
                                0D0C  4264
     53   BC A5   D0             0D0C  4265          MOVL    CDRP$L_UCB(R5), R3              ; Get UCB address.
08   C0 A5   06   00             0D10  4266          CMPZV   #IO$V_FCODE, #IO$S_FCODE, -     ; Branch if this was not a
                                0D16  4267                  CDRP$Q_FUNC(R5), #IO$_PACKACK   ; PACKACK function.
          08    12              0D16  4268          BNEQ    14$
    03 68 A3   0C               0D18  4269          BBCC    #UCB$V_MSCP_PKACK, -            ; Clear packack flag and
                                0D1D  4270                  UCB$W_DEVSTS(R3), 14$           ; branch if it was clear.
        56 A3   B7              0D1D  4271          DECW    UCB$W_RWAITCNT(R3)              ; Else, decrement wait count.
                                0D20  4272
1A   C0 A5   06   00             0D20  4273 14$:     CMPZV   #IO$V_FCODE, #IO$S_FCODE, -     ; Branch if this was not a
                                0D26  4274                  CDRP$Q_FUNC(R5), #IO$_SETCHAR   ; SETCHAR function.
          10    12              0D26  4275          BNEQ    17$
23   C0 A5   06   00             0D28  4276          CMPZV   #IO$V_FCODE, #IO$S_FCODE, -     ; Branch if this was not a
                                0D2E  4277                  CDRP$Q_FUNC(R5), #IO$_SETMODE   ; SETMODE function.
          08    12              0D2E  4278          BNEQ    17$
```

```
                03 68 A3    02    E5  0D30  4279          BBCC    #UCB$V_TU_SEQNOP, -                 ; Clear seq. NOP flag and
                                      0D35  4280                  UCB$W_DEVSTS(R3), 17$               ; branch if it was clear.
                        56  A3    B7  0D35  4281          DECW    UCB$W_RWAITCNT(R3)                  ; Else, decrement wait count.
                                      0D38  4282
                        55    DD    0D38  4283 17$:       PUSHL   R5                                  ; Save CDRP address.
                     55       53    D0  0D3A  4284          MOVL    R3, R5                              ; Copy UCB address.
                00000000'GF    16    0D3D  4285          JSB     G^SCS$UNSTALLUCB                    ; Possibly unstall the UCB.
                        55  8ED0    0D43  4286          POPL    R5                                  ; Restore CDRP address.
                                      0D46  4287
                04 CA A5    00    E1  0D46  4288          BBC     #IRP$V_BUFIO, -                     ; Is this a direct I/O request?
                                      0D4B  4289                  CDRP$W_STS(R5), 20$                 ; Branch if direct I/O.
                   CA A5    02    AA  0D4B  4290          BICW    #IRP$M_FUNC, CDRP$W_STS(R5)         ; Else, clear buffered read bit.
                                      0D4F  4291
             50 00000000'GF    9E    0D4F  4292 20$:       MOVAB   G^IOC$GL_PSBL, R0                   ; Get I/O post queue tail ptr.
                00 B0    A0 A5    0E  0D56  4293          INSQUE  CDRP$L_IOQFL(R5), @(R0)             ; Insert CDRP on post queue.
                                      0D5B  4294          SOFTINT #IPL$_IOPOST                        ; Request post processing serv.
                                      0D5E  4295
                        3C    BA    0D5E  4296          POPR    #^M<R2,R3,R4,R5>                    ; Restore registers.
                              05    0D60  4297          RSB                                         ; Return.
                                      0D61  4298
                                      0D61  4299 999$:      BUG_CHECK MSCPCLASS, FATAL                ; Attempt to insert non-CDRP in
                                      0D65  4300                                                      ; the I/O post processing queue.
```

```
                     0D65  4302          .SBTTL  DUTU$KILL_THIS_THREAD - Fix thread caught by async. conn. failure
                     0D65  4303 ;++
                     0D65  4304 ;
                     0D65  4305 ; DUTU$KILL_THIS_THREAD - Fix thread caught by asynchronous connection failure
                     0D65  4306 ;
                     0D65  4307 ; Functional Description:
                     0D65  4308 ;
                     0D65  4309 ;       This is an internal routine that is either jumped (BRW) to or called
                     0D65  4310 ;       (BSBW).
                     0D65  4311 ;
                     0D65  4312 ;       It is jumped to by a thread that has suffered an allocation failure
                     0D65  4313 ;       indicating that its CONNECTION has failed.  This should only happen
                     0D65  4314 ;       when the CONNECTION fails asynchronously with respect to the class
                     0D65  4315 ;       driver and the cause of the CONNECTION failure (e.g. power failure)
                     0D65  4316 ;       interrupts this very driver thread in mid execution.
                     0D65  4317 ;
                     0D65  4318 ;       This code is called when the class driver determines that the
                     0D65  4319 ;       intelligent controller is acting in an "insane" manner and that the
                     0D65  4320 ;       class driver therefore should bring down the CONNECTION and
                     0D65  4321 ;       re-synchronize (i.e. re-CONNECT) with the controller.
                     0D65  4322 ;
                     0D65  4323 ;       In either case, the CDRP is placed at the tail of the connection
                     0D65  4324 ;       outstanding requests queue, CDDB$L_CDRPQBL.  This queue holds all
                     0D65  4325 ;       those CDRP's with outstanding requests in the controller plus those
                     0D65  4326 ;       CDRP's of "killed" driver threads.  What all these CDRP's have in
                     0D65  4327 ;       common is that they are NOT on any resource wait queues.  After
                     0D65  4328 ;       queueing the CDRP, this routine does an RSB which returns to either to
                     0D65  4329 ;       caller's caller if entry was made via a BRW or to caller if entry was
                     0D65  4330 ;       made via a BSBW.
                     0D65  4331 ;
                     0D65  4332 ; Inputs:
                     0D65  4333 ;
                     0D65  4334 ;       R3      UCB address
                     0D65  4335 ;       R5      CDRP address
                     0D65  4336 ;
                     0D65  4337 ; Implicit Inputs:
                     0D65  4338 ;
                     0D65  4339 ;       UCB$L_CDDB(R3)          CDDB address
                     0D65  4340 ;       CDDB$L_CDRPQBL(cddb)    outstanding request queue backlink
                     0D65  4341 ;
                     0D65  4342 ; Outputs:
                     0D65  4343 ;
                     0D65  4344 ;       R1 is destroyed.
                     0D65  4345 ;       All other registers are preserved.
                     0D65  4346 ;--
                     0D65  4347
                     0D65  4348 DUTU$KILL_THIS_THREAD::
                     0D65  4349
51   00BC C3    D0   0D65  4350          MOVL    UCB$L_CDDB(R3), R1              ; Get CDDB address
  04 B1    65    0E   0D6A  4351          INSQUE  (R5), @CDDB$L_CDRPQBL(R1)       ; Insert CDRP onto tail of queue
                     0D6E  4352                                                  ; of CDRP's sent to the port.
                05   0D6E  4353          RSB                                     ; Return to caller or
                     0D6F  4354                                                  ; caller's caller.
```

```
                                0D6F  4356          .SBTTL  DUTU$CHECK_RWAITCNT - Validate UCB$W_RWAITCNT
                                0D6F  4357 ;++
                                0D6F  4358 ;
                                0D6F  4359 ; DUTU$CHECK_RWAITCNT - Validate UCB$W_RWAITCNT
                                0D6F  4360 ;
                                0D6F  4361 ; Functional Description:
                                0D6F  4362 ;
                                0D6F  4363 ;       This routine compares UCB$W_RWAITCNT against a computed value based
                                0D6F  4364 ;       upon information available as to the reasons that RWAITCNT might be
                                0D6F  4365 ;       bumped.  If the two values compare, control is returned to the caller.
                                0D6F  4366 ;       If they do not compare, an INCONSTATE bug check is generated.
                                0D6F  4367 ;
                                0D6F  4368 ;       This routine does not test for SCS wait states as possible
                                0D6F  4369 ;       incrementers of UCB$W_RWAITCNT.  Therefore, it MUST NOT be called when
                                0D6F  4370 ;       a possibility exists for a thread to be in an SCS wait state.
                                0D6F  4371 ;
                                0D6F  4372 ; Inputs:
                                0D6F  4373 ;
                                0D6F  4374 ;       R5      UCB address
                                0D6F  4375 ;
                                0D6F  4376 ; Outputs:
                                0D6F  4377 ;
                                0D6F  4378 ;       All registers preserved.
                                0D6F  4379 ;--
                                0D6F  4380
                                0D6F  4381 DUTU$CHECK_RWAITCNT::
                                0D6F  4382
                7E    50   7D   0D6F  4383          MOVQ    R0, -(SP)                       ; Save corrupted registers.
                      50   D4   0D72  4384          CLRL    R0                              ; Init accumulator.
                                0D74  4385
             02 68 A5   0A   E1   0D74  4386          BBC     #UCB$V_MSCP_WAITBMP, -          ; Check for wait count
                                0D79  4387                  UCB$W_DEVSTS(R5), 10$           ; explicitly bumped.
                      50   D6   0D79  4388          INCL    R0                              ; If so, bump accumulator.
                                0D7B  4389
             02 68 A5   08   E1   0D7B  4390 10$:     BBC     #UCB$V_MSCP_MNTVERIP, -         ; Check for wait count bumped
                                0D80  4391                  UCB$W_DEVSTS(R5), 12$           ; by mount verification.
                      50   D6   0D80  4392          INCL    R0                              ; If so, bump accumulator.
                                0D82  4393
             02 68 A5   0C   E1   0D82  4394 12$:     BBC     #UCB$V_MSCP_PKACK, -            ; Check for wait count bumped
                                0D87  4395                  UCB$W_DEVSTS(R5), 20$           ; by packack in progress.
                      50   D6   0D87  4396          INCL    R0                              ; If so, bump accumulator.
                                0D89  4397
                                0D89  4398 20$:     .WEAK   DU$TEST_HIRT_RWAITCNT
       51    00000000'GF   9E   0D89  4399          MOVAB   G^DU$TEST_HIRT_RWAITCNT, R1     ; Get routine address for HIRT
                      02   13   0D90  4400          BEQL    30$                             ; check and branch if none.
                      61   16   0D92  4401          JSB     (R1)                            ; Else, perform HIRT check.
                                0D94  4402
                40 A5   02   91   0D94  4403 30$:     CMPB    #DC$_TAPE, UCB$B_DEVCLASS(R5)   ; Is this a tape?
                      07   12   0D98  4404          BNEQ    40$                             ; Branch if not a tape.
             02 68 A5   02   E1   0D9A  4405          BBC     #UCB$V_TU_SEQNOP, -             ; Is a sequential NOP in
                                0D9F  4406                  UCB$W_DEVSTS(R5), 40$           ; progress:  branch if no.
                      50   D6   0D9F  4407          INCL    R0                              ; Else, bump accumulator.
                                0DA1  4408
                56 A5   50   B1   0DA1  4409 40$:     CMPW    R0, UCB$W_RWAITCNT(R5)          ; Check for correct RWAITCNT.
                      04   12   0DA5  4410          BNEQ    99$                             ; Branch if no match.
                                0DA7  4411
                   50   8E   7D   0DA7  4412          MOVQ    (SP)+, R0                       ; Restore registers.
```

```
05  0DAA  4413          RSB                                     ; Return.
    0DAB  4414
    0DAB  4415 99$:     BUG_CHECK MSCPCLASS, FATAL              ; RWAITCNT test failed.
```

```
ODAF  4417          .SBTTL  DUTU$LOG_IVCMD - Error log an invalid MSCP command
ODAF  4418 ;++
ODAF  4419 ;
ODAF  4420 ; DUTU$LOG_IVCMD - Error log an invalid MSCP command
ODAF  4421 ;
ODAF  4422 ; Functional Description:
ODAF  4423 ;
ODAF  4424 ;       Called via the IVCMD_BEGIN macro in response to receipt of an "invalid
ODAF  4425 ;       command" MSCP end-packet status, this routine produces an error log
ODAF  4426 ;       entry containing both the entire MSCP end-packet and the MSCP command
ODAF  4427 ;       which produced that end-packet.
ODAF  4428 ;
ODAF  4429 ;       Saving every command packet sent out for possible insertion in the
ODAF  4430 ;       error log in the unlikely event of an "invalid command" error is
ODAF  4431 ;       highly inefficient.  Therefore, in the rare cases where an "invalid
ODAF  4432 ;       command" error is reported, the class driver must "retrace its steps"
ODAF  4433 ;       to produce a replicate of the MSCP command which caused the error.
ODAF  4434 ;       This is accomplished by producing a co-routine execution thread in
ODAF  4435 ;       which the class driver executes most (if not all) the same instructions
ODAF  4436 ;       it executed to build the original MSCP command packet.  The only
ODAF  4437 ;       instructions which cannot be executed in this co-routine thread are
ODAF  4438 ;       those instructions which might attempt to suspend the fork execution
ODAF  4439 ;       thread (e.g. allocating mapping resources).  The invalid command
ODAF  4440 ;       co-routine thread cannot break IPL synchronization.  Then, just before
ODAF  4441 ;       the duplicate command packet would be sent to the MSCP server, control
ODAF  4442 ;       is returned to this routine, which places the information in the error
ODAF  4443 ;       log.
ODAF  4444 ;
ODAF  4445 ;       Three macros are provided for class drivers to use in handling
ODAF  4446 ;       "invalid command" errors:
ODAF  4447 ;
ODAF  4448 ;       IVCMD_BEGIN     begin processing an invalid command and start the
ODAF  4449 ;                       the co-routine thread
ODAF  4450 ;
ODAF  4451 ;       IF_IVCMD        branch if within an invalid command co-routine thread
ODAF  4452 ;
ODAF  4453 ;       IVCMD_END       end an invalid command co-routine thread, make the
ODAF  4454 ;                       error log entry, and return to normal processing
ODAF  4455 ;
ODAF  4456 ;       For a class driver, the invalid command processing style might look
ODAF  4457 ;       something like this:
ODAF  4458 ;
ODAF  4459 ;       START_func:
ODAF  4460 ;               ; Setup packet for command func.
ODAF  4461 ;               IF_IVCMD then=func_IVCMD_END
ODAF  4462 ;               SEND_MSCP_MSG
ODAF  4463 ;               DO_ACTION       xxxxx
ODAF  4464 ;                       .
ODAF  4465 ;               ACTION_ENTRY    ICMD, SS$_CNTRLERR, func_IVCMD
ODAF  4466 ;                       .
ODAF  4467 ;               ACTION_ENTRY    END_TABLE
ODAF  4468 ;                       .
ODAF  4469 ;       func_IVCMD:
ODAF  4470 ;               IVCMD_BEGIN
ODAF  4471 ;               BRB     START_func
ODAF  4472 ;       func_IVCMD_END:
ODAF  4473 ;               IVCMD_END
```

```
                        BRW     FUNCTION_EXIT

        Note:  all invalid command processing occurs out-of-line with respect
        to the main execution thread.  Invalid command processing is not the
        normal case and rightfully should be processed outside the mainline
        code path.  IF_IVCMD contains a single, highly-optimized instruction.

        Somtimes it will be necessary to repeat the start I/O dispatch off of
        IRP$W_FUNC in order to put the correct MSCP opcode and modifiers into
        the duplicate MSCP packet.  For this purpose, both class drivers
        define a label of the form xx_BEGIN_IVCMD (where xx is either DU or TU,
        as appropriate).  This label has the implicit guarantee that nowhere
        between it and the beginning of the function specific start I/O
        routine are there any instructions which cannot be executed in an
        "invalid command" co-routine thread.

Discription of Macros:

IVCMD_BEGIN

Parameters: None.

Inputs:

        R0      SS$_ status code for the IOSB after the error is logged
        R1      scratch
        R2      MSCP end-packet address
        R3      UCB address
        R4      PDT address
        R5      CDRP address

Outputs (beginning of the co-routine thread):

        R0      scratch
        R1      scratch
        R2      Duplicate MSCP command packet address (for forming the MSCP
                command which caused the error)
        R3      UCB address (same as input to IVCMD_BEGIN)
        R4      PDT address (same as input to IVCMD_BEGIN)
        R5      CDRP address (same as input to IVCMD_BEGIN)

IF_IVCMD

Parameters:

        then    label to receive control when execution is within an invalid
                command co-routine thread

Inputs:

        R5      CDRP address (same as input/output to IVCMD_BEGIN)

Outputs: None.

IVCMD_END

Parameters: None.
```

```
ODAF  4531 ;
ODAF  4532 ; Inputs:
ODAF  4533 ;
ODAF  4534 ;       R0      scratch
ODAF  4535 ;       R1      scratch
ODAF  4536 ;       R2      Duplicate MSCP command pkt. address (as output from IVCMD_BEGIN)
ODAF  4537 ;       R3      UCB address (same as input/output to IVCMD_BEGIN)
ODAF  4538 ;       R4      PDT address (same as input/output to IVCMD_BEGIN)
ODAF  4539 ;       R5      CDRP address (same as input/output to IVCMD_BEGIN)
ODAF  4540 ;
ODAF  4541 ; Outputs:
ODAF  4542 ;
ODAF  4543 ;       R0      SS$_ status code for the IOSB (same as input to IVCMD_BEGIN)
ODAF  4544 ;       R1      LBC ==> CDRP$V_ERLIP clear when IVCMD_BEGIN called
ODAF  4545 ;               LBS ==> CDRP$V_ERLIP set when IVCMD_BEGIN called
ODAF  4546 ;       R2      MSCP end-packet address (same as input to IVCMD_BEGIN)
ODAF  4547 ;       R3      UCB address (same as input to IVCMD_BEGIN)
ODAF  4548 ;       R4      PDT address (same as input to IVCMD_BEGIN)
ODAF  4549 ;       R5      CDRP address (same as input to IVCMD_BEGIN)
ODAF  4550 ;
ODAF  4551 ; Implicit Outputs:
ODAF  4552 ;
ODAF  4553 ;       CDRP$L_DUTUFLAGS(R5)    CDRP$V_ERLIP set
ODAF  4554 ;
ODAF  4555 ; Side Effects:
ODAF  4556 ;
ODAF  4557 ;       While the invalid command co-routine thread is executing, a relatively
ODAF  4558 ;       complex stack structure exists below the stack section used within the
ODAF  4559 ;       co-routine thread itself.  As much as anything else, this complex
ODAF  4560 ;       stack structure prohibits the co-routine thread from being fork-
ODAF  4561 ;       suspended or from breaking IPL synchronization.  Generally speaking,
ODAF  4562 ;       the stack structure looks like this:
ODAF  4563 ;
ODAF  4564 ;       |-------------------------------------------------------------------|
ODAF  4565 ;       |                                                                   |
ODAF  4566 ;       \    stack space for use by invalid command co-routine thread       \
ODAF  4567 ;       |                                                                   |
ODAF  4568 ;       |-------------------------------------------------------------------|
ODAF  4569 ;       |                 return address used by IVCMD_END                  |  :(SP)
ODAF  4570 ;       |-------------------------------------------------------------------|
ODAF  4571 ;       |  zero (used to determine that control was correctly returned)     |
ODAF  4572 ;       |-------------------------------------------------------------------|
ODAF  4573 ;       |                                                                   |  :(R7)
ODAF  4574 ;       \               invalid command MSCP end packet                     \
ODAF  4575 ;       |                                                                   |
ODAF  4576 ;       |-------------------------------------------------------------------|
ODAF  4577 ;       |                                                                   |  :(R2)
ODAF  4578 ;       \             invalid command MSCP command packet                   \
ODAF  4579 ;       |                                                                   |
ODAF  4580 ;       |-------------------------------------------------------------------|
ODAF  4581 ;       |                                                                   |
ODAF  4582 ;       \          other stacklocals used by DUTU$LOG_IVCMD                 \
ODAF  4583 ;       |                                                                   |
ODAF  4584 ;       |-------------------------------------------------------------------|
ODAF  4585 ;       |                 saved SS$_ status value (saved R0)                |
ODAF  4586 ;       |-------------------------------------------------------------------|
ODAF  4587 ;       |     saved CDRP$V_ERLIP value in low bit (returned in R1)          |
```

```
                     0DAF  4588 ;       |-----------------------------------------------------------|
                     0DAF  4589 ;       |          saved MSCP end-packet address (saved R2)          |
                     0DAF  4590 ;       |-----------------------------------------------------------|
                     0DAF  4591 ;       |              saved UCB address (saved R3)                  |
                     0DAF  4592 ;       |-----------------------------------------------------------|
                     0DAF  4593 ;       |              saved PDT address (saved R4)                  |
                     0DAF  4594 ;       |-----------------------------------------------------------|
                     0DAF  4595 ;       |              saved CDRP address (saved R5)                 |
                     0DAF  4596 ;       |-----------------------------------------------------------|
                     0DAF  4597 ;       |                    saved R7                               |
                     0DAF  4598 ;       |-----------------------------------------------------------|
                     0DAF  4599 ;       |  saved return address from IVCMD_BEGIN and later IVCMD_END |
                     0DAF  4600 ;       |-----------------------------------------------------------|
                     0DAF  4601 ;         ' R7 is not an explicit input/output for any of the invalid
                     0DAF  4602 ;           command processing logic.  It is mentioned here simply as
                     0DAF  4603 ;           a point of information.
                     0DAF  4604 ;--
                     0DAF  4605
                     0DAF  4606         ; Define the stack offsets for which R7 will act as base.
                     0DAF  4607         ;
                     0DAF  4608         ; N.B. these offsets assume the following initialization sequence:
                     0DAF  4609         ;
                     0DAF  4610         ; entry:
                     0DAF  4611         ;
                     0DAF  4612         ;       PUSHR   #^M<R0,R1,R2,R3,R4,R5,R7>
                     0DAF  4613         ;       MOVAB   -IVCMD_WORK(SP), SP
                     0DAF  4614         ;       MOVL    SP, R7
                     0DAF  4615         ;
                     0DAF  4616
 00000030            0DAF  4617         IVCMD_ENDSIZ = MSCP$K_MXCMDLEN + 12
                     0DAF  4618
                     0DAF  4619         $OFFSET 0, POSITIVE, <-                     ; Define R7 based offsets:
                     0DAF  4620                 <IVCMD_ENDMSG, IVCMD_ENDSIZ>, -         ; End message buffer
                     0DAF  4621                 <IVCMD_ORGMSG, MSCP$K_MXCMDLEN>, -      ; Original message buf.
                     0DAF  4622                 <IVCMD_MSGLEN, 0>, -            ; Size of both message buffers.
                     0DAF  4623                 <IVCMD_ALIGN, <<<IVCMD_MSGLEN+3>&^c3>-IVCMD_MSGLEN>>, -
                     0DAF  4624                 -
                     0DAF  4625                 - ; Add local working storage after this line.
                     0DAF  4626                 -
                     0DAF  4627                 <IVCMD_WORK, 0>, -              ; Total stacklocal size.
                     0DAF  4628                 -                               ; Saved registers etc.:
                     0DAF  4629                 <SVSTATUS>, -                   ; R0, SS$_ status code
                     0DAF  4630                 <SVERLIP>, -                    ; R1, saved ERLIP flag
                     0DAF  4631                 <MSCP_BUF>, -                   ; R2, MSCP message buffer.
                     0DAF  4632                 <SVUCB>, -                      ; R3, UCB address.
                     0DAF  4633                 <SVPDT>, -                      ; R4, PDT address.
                     0DAF  4634                 <SVCDRP>, -                     ; R5, CDRP address.
                     0DAF  4635                 <SVR7>, -                       ; R7, who knows or cares.
                     0DAF  4636                 <SVRET> -                       ; Return address.
                     0DAF  4637                 >
                     0000       IVCMD_ENDMSG:
                     0030       IVCMD_ORGMSG:
                     0054       IVCMD_MSGLEN:
                     0054       IVCMD_ALIGN:
                     0054       IVCMD_WORK:
                     0054       SVSTATUS:
                     0058       SVERLIP:
```

```
                                005C        MSCP_BUF:
                                0060        SVUCB:
                                0064        SVPDT:
                                0068        SVCDRP:
                                006C        SVR7:
                                0070        SVRET:
                                0DAF  4638
                                0DAF  4639  DUTU$LOG_IVCMD::
                                0DAF  4640
                00BF 8F   BB    0DAF  4641          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R7>          ; Save registers.
             5E   AC AE   9E    0DB3  4642          MOVAB   -IVCMD_WORK(SP), SP                 ; Make stacklocal space.
                57   5E   D0    0DB7  4643          MOVL    SP, R7                              ; Setup stacklocal pointer.
                                0DBA  4644
          67    62   30   28    0DBA  4645          MOVC3   #IVCMD_ENDSIZ, (R2), -              ; Copy the MSCP end-packet to
                                0DBE  4646                  IVCMD_ENDMSG(R7)                    ; the error log work area.
                                0DBE  4647          ASSUME  SVCDRP EQ <SVPDT + 4>
             54   64 A7   7D    0DBE  4648          MOVQ    SVPDT(R7), R4                       ; Restore PDT and CDRP addrs.
             53   60 A7   D0    0DC2  4649          MOVL    SVUCB(R7), R3                       ; Restore UCB address.
                                0DC6  4650
    58 A7   40 A5   01   02    EF  0DC6  4651          EXTZV   #CDRP$V_ERLIP, #1, -                ; Save the ERLIP bit.
                                0DCD  4652                  CDRP$L_DUTUFLAGS(R5), SVERLIP(R7)
          1C A5   30 A7   9E    0DCD  4653          MOVAB   IVCMD_ORGMSG(R7), -                 ; Initialize the duplicate
                                0DD2  4654                  CDRP$L_MSG_BUF(R5)                  ; MSCP command pkt. address.
                                0DD2  4655          INIT_MSCP_MSG ucb=(R3)                      ; Initialize the pkt. itself.
                                0DD5  4656                                                      ; R2 now has dup. pkt. addr.
          1C A5   5C A7   D0    0DD5  4657          MOVL    MSCP_BUF(R7), -                     ; Restore correct MSCP end
                                0DDA  4658                  CDRP$L_MSG_BUF(R5)                  ; packet address in CDRP.
                                0DDA  4659          ASSUME  CDRP$V_IVCMD EQ 8
             41 A5   01   88    0DDA  4660          BISB    #<CDRP$M_IVCMD @ -8>, -             ; Signal that an invalid
                                0DDE  4661                  CDRP$L_DUTUFLAGS+1(R5)              ; command co-routine is active.
                                0DDE  4662
                                0DDE  4663          ; The JSB which follows initiates the co-routine thread.
                                0DDE  4664          ; IVCMD_END will return control to the instruction following the JSB.
                                0DDE  4665          ; The zero longword pushed onto the stack is used below to determine
                                0DDE  4666          ; whether control was returned via IVCMD_END or a RSB.
                                0DDE  4667
                7E   D4         0DDE  4668          CLRL    -(SP)                               ; Push signal longword.
             70 B7   16         0DE0  4669          JSB     @SVRET(R7)                          ; Begin co-routine thread.
                                0DE3  4670
                                0DE3  4671          ; The following test attempts to insure that the co-routine thread
                                0DE3  4672          ; started by the JSB above has properly returned here via IVCMD_END.
                                0DE3  4673          ; With luck, this test allows timely detection of a bug in the class
                                0DE3  4674          ; driver, so that we can drop our cookies on the floor neatly.
                                0DE3  4675
                50 8ED0         0DE3  4676          POPL    R0                                  ; Get final return address.
                21   13         0DE6  4677          BEQL    999$                                ; Branch if signal lw. poped.
          70 A7 50   D0         0DE8  4678          MOVL    R0, SVRET(R7)                       ; Save final return address.
                                0DEC  4679          ASSUME  CDRP$V_IVCMD EQ 8
          41 A5   01   8A       0DEC  4680          BICB    #<CDRP$M_IVCMD @ -8>, -             ; Clear the invalid command
                                0DF0  4681                  CDRP$L_DUTUFLAGS+1(R5)              ; co-routine is active flag.
                                0DF0  4682
                                0DF0  4683          ; Make invalid command error log entry.
                                0DF0  4684
             50   07   9A       0DF0  4685          MOVZBL  #EMB$C_IVCMD, R0                    ; Signal type of logged message.
          51   54 8F   9A       0DF3  4686          MOVZBL  #IVCMD_MSGLEN, R1                   ; Pass message length.
             52   67   9E       0DF7  4687          MOVAB   IVCMD_ENDMSG(R7), R2                ; Pass message base address.
00000000'GF   16         0DFA  4688          JSB     G^ERL$LOGMESSAGE                    ; Call to error log message.
```

```
                     0E00  4689
5E   58 AE    9E    0E00  4690          MOVAB   IVCMD_WORK+4(SP), SP            ; Pop stacklocal zero longword.
   00BF 8F    BA    0E04  4691          POPR    #^M<R0,R1,R2,R3,R4,R5,R7>       ; Restore registers & outputs.
              05    0E08  4692          RSB                                     ; Return.
                    0E09  4693
                    0E09  4694 999$:    BUG_CHECK MSCPCLASS, FATAL              ; Control was not returned
                    0E0D  4695                                                  ; from the invalid_command
                    0E0D  4696                                                  ; co-routine via a IVCMD_END.
```

```
                                0E0D  4698          .SBTTL  DUTU$DODAP - Do Determine Access Paths Processing
                                0E0D  4699 ;++
                                0E0D  4700 ;
                                0E0D  4701 ; DUTU$DODAP - Do Determine Access Paths Processing
                                0E0D  4702 ;
                                0E0D  4703 ; Functional Description:
                                0E0D  4704 ;
                                0E0D  4705 ;       This routine supervises the sending of Determine Access Path (DAP)
                                0E0D  4706 ;       commands for the purpose of discovering the current topology of the
                                0E0D  4707 ;       MSCP units (either disks or tapes) accesible to this processor.  These
                                0E0D  4708 ;       commands make us aware of alternate (not currently accessible) paths
                                0E0D  4709 ;       to mounted units.
                                0E0D  4710 ;
                                0E0D  4711 ; Inputs:
                                0E0D  4712 ;
                                0E0D  4713 ;       R1      CDDB address
                                0E0D  4714 ;
                                0E0D  4715 ; Outputs:
                                0E0D  4716 ;
                                0E0D  4717 ;       Registers R0 through R5 are modified.
                                0E0D  4718 ;--
                                0E0D  4719
                                0E0D  4720 ; Number of passes through DODAP before each DAP processing thread is started.
                                0E0D  4721
                     0000000A   0E0D  4722 DAP_COUNT = 10
                                0E0D  4723
                                0E0D  4724 DUTU$DODAP::
                                0E0D  4725
 57 12 A1    0A    E2           0E0D  4726          BBSS    #CDDB$V_DAPBSY, -                       ; Branch if DAP CDRP is in
                                0E12  4727                  CDDB$W_STATUS(R1), 90$                  ; use and set in use flag.
       39 A1    97              0E12  4728          DECB    CDDB$B_DAPCOUNT(R1)                     ; Count number of passes through
          4E    18              0E15  4729          BGEQ    80$                                     ; here before going again.
    39 A1    0A    90           0E17  4730          MOVB    #DAP_COUNT, -                           ; Refresh passes counter.
                                0E1B  4731                  CDDB$B_DAPCOUNT(R1)
                                0E1B  4732
    55    54 A1    D0           0E1B  4733          MOVL    CDDB$L_DAPCDRP(R1), R5                  ; Get DAP CDRP address.
    54    14 A1    D0           0E1F  4734          MOVL    CDDB$L_PDT(R1), R4                      ; Get PDT address.
    53    84 A1    9E           0E23  4735          MOVAB   <CDDB$L_UCBCHAIN -                      ; Setup "previous" UCB address
                                0E27  4736                  -UCB$L_CDDB_LINK>(R1), R3               ; in R3.
                                0E27  4737          ALLOC_RSPID                                     ; Allocate a RSPID.
                                0E2D  4738
  53    00C4 C3    D0           0E2D  4739 10$:     MOVL    UCB$L_CDDB_LINK(R3), R3                 ; Link to next UCB.
          24    13              0E32  4740          BEQL    50$                                     ; Branch if no more UCBs.
 F4 64 A3    0B    E1           0E34  4741          BBC     #UCB$V_VALID, -                         ; If this UCB is not VALID (or
                                0E39  4742                  UCB$L_STS(R3), 10$                      ; MSCP online), then skip it.
    BC A5    53    D0           0E39  4743          MOVL    R3, CDRP$L_UCB(R5)                      ; Put UCB in DAP CDRP.
                                0E3D  4744          ALLOC_MSG_BUF                                   ; Allocate a message buffer.
       15 50    E9              0E40  4745          BLBC    R0, 50$                                 ; Branch if connection failure.
                                0E43  4746          INIT_MSCP_MSG ucb=(R3)                          ; Initialize MSCP packet.
    08 A2    0B    90           0E46  4747          MOVB    #MSCP$K_OP_DTACP, -                     ; Set DAP opcode.
                                0E4A  4748                  MSCP$B_OPCODE(R2)
                                0E4A  4749          SEND_MSCP_MSG                                   ; Send message to MSCP server.
                                0E4D  4750          DEALLOC_MSG_BUF                                 ; Deallocate response message.
                                0E50  4751          RECYCL_RSPID                                    ; Recycle RSPID for reuse.
          D5    11              0E56  4752          BRB     10$                                     ; Loop through all UCBs.
                                0E58  4753
                                0E58  4754 50$:     DEALLOC_RSPID                                   ; Deallocate DAP RSPID.
```

```
                            0E5E  4755              PERMCDRP_TO_CDDB cdrp=R5, cddb=R1        ; Get CDDB address.
                            0E65  4756 80$:         ASSUME  CDDB$V_DAPBSY GE 8
                 13 A1  04  8A 0E65  4757           BICB    #<CDDB$M_DAPBSY @ -8>, -         ; Clear DAP CDRP in use flag.
                            0E69  4758                      CDDB$W_STATUS+1(R1)
                            05 0E69  4759 90$:      RSB                                      ; Kill this fork thread.
```

```
                                0E6A  4761          .SBTTL  DUTU$SEND_DUPLICATE_UNIT - Send duplicate unit message to operator
                                0E6A  4762
                                0E6A  4763 ;++
                                0E6A  4764 ;
                                0E6A  4765 ; DUTU$SEND_DUPLICATE_UNIT - Send duplicate unit message to operator
                                0E6A  4766 ;
                                0E6A  4767 ; Functional Description:
                                0E6A  4768 ;
                                0E6A  4769 ;       A message indicating the detection of a duplicate MSCP unit number
                                0E6A  4770 ;       is sent to OPCOM.
                                0E6A  4771 ;
                                0E6A  4772 ; Inputs:
                                0E6A  4773 ;
                                0E6A  4774 ;       R3      address of the UCB for which a MSCP server detected a
                                0E6A  4775 ;               duplicate unit number
                                0E6A  4776 ;
                                0E6A  4777 ; Outputs:
                                0E6A  4778 ;
                                0E6A  4779 ;       All registers preserved.
                                0E6A  4780 ;--
                                0E6A  4781
                                0E6A  4782 DUTU$SEND_DUPLICATE_UNIT::
                                0E6A  4783
                          3F    BB  0E6A  4784          PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Save registers.
                       55 53    D0  0E6C  4785          MOVL    R3, R5                          ; Jockey UCB addr. to right reg.
                    54    58 8F    9A  0E6F  4786          MOVZBL  #MSG$_DUPUNITNO, R4             ; Get message number.
                53  00000000'GF    9E  0E73  4787          MOVAB   G^SYS$GL_OPRMBX, R3             ; Get OPCOM's mailbox address.
                    00000000'GF    16  0E7A  4788          JSB     G^EXE$SNDEVMSG                  ; Send the message.
                          3F    BA  0E80  4789          POPR    #^M<R0,R1,R2,R3,R4,R5>          ; Restore registers.
                                05  0E82  4790          RSB
```

```
                                0E83  4792          .SBTTL  "Register" Dump Routines
                                0E83  4793          .SBTTL  o DUTU$DUMP_COMMAND - Copy MSCP command to diagnostic buffer
                                0E83  4794
                                0E83  4795 ;++
                                0E83  4796 ;
                                0E83  4797 ; DUTU$DUMP_COMMAND - Copy MSCP command to diagnostic buffer
                                0E83  4798 ;
                                0E83  4799 ; Functional Description:
                                0E83  4800 ;
                                0E83  4801 ;       Copy the contents of the MSCP Command to the diagnostic buffer whose
                                0E83  4802 ;       address is pointed to by CDRP$L_DIAGBUF(R5).  This is the MSCP message
                                0E83  4803 ;       which will be sent to the remote server.
                                0E83  4804 ;
                                0E83  4805 ; Inputs:
                                0E83  4806 ;       R5 CDRP address
                                0E83  4807 ;--
                                0E83  4808
                                0E83  4809 DUTU$DUMP_COMMAND::
                                0E83  4810
                          3F    BB  0E83  4811          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers.
                    50    EC    B5  D0  0E85  4812          MOVL    @CDRP$L_DIAGBUF(R5), R0 ; Get diagnostic buffer address.
              14    A0    1C    A5  24  28  0E89  4813          MOVC3   #MSCP$K_MXCMDLEN, -     ; Copy the entire command to the
                                0E8F  4814                          CDRP$L_MSG_BUF(R5), -   ;  diagnostic buffer.
                                0E8F  4815                          20(R0)
                          3F    BA  0E8F  4816          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                                05  0E91  4817          RSB                             ; Return.
```

```
                                0E92  4819          .SBTTL  o DUTU$DUMP_ENDMESSAGE - Copy MSCP end message to diagnostic buffer
                                0E92  4820
                                0E92  4821 ;++
                                0E92  4822 ;
                                0E92  4823 ; DUTU$DUMP_ENDMESSAGE - Copy MSCP end message to diagnostic buffer
                                0E92  4824 ;
                                0E92  4825 ; Functional Description:
                                0E92  4826 ;
                                0E92  4827 ;       Copy the contents of the MSCP end message to the diagnostic buffer
                                0E92  4828 ;       whose address is pointed to by CDRP$L_DIAGBUF(R5).  This is the MSCP
                                0E92  4829 ;       end message returned to the class driver by the remote server.
                                0E92  4830 ;
                                0E92  4831 ; Inputs:
                                0E92  4832 ;
                                0E92  4833 ;       R1 END Message length
                                0E92  4834 ;       R2 END Message address
                                0E92  4835 ;       R5 CDRP address
                                0E92  4836 ;--
                                0E92  4837
                                0E92  4838 DUTU$DUMP_ENDMESSAGE::
                                0E92  4839
                          3F    BB  0E92  4840          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers.
                       50   EC  B5  0E94  4841          MOVL    @CDRP$L_DIAGBUF(R5), R0 ; Get diagnostic buffer address.
                08 A0  00000000'GF  7D  0E98  4842          MOVQ    G^EXE$GQ_SYSTIME, 8(R0) ; Save ending time.
                          10 A0  D4  0EA0  4843          CLRL    16(R0)                  ; Clear retry counts.
           38 A0  30  00  62  51  2C  0EA3  4844          MOVC5   R1, (R2), #0, -         ; Copy end message to diagnostic
                                  0EAA  4845                  #MSCP$K_LEN, -          ;  buffer zero filling for short
                                  0EAA  4846                  MSCP$K_MXCMDLEN+20(R0)  ;  end messages.
                          3F    BA  0EAA  4847          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                                05  0EAC  4848          RSB                             ; Return.
```

```
                                0EAD  4850          .SBTTL  DUTU$FILL_MSCP_MSG - Zero fill a partial MSCP message
                                0EAD  4851 ;++
                                0EAD  4852 ;
                                0EAD  4853 ; DUTU$FILL_MSCP_MSG - Zero fill a partial MSCP message
                                0EAD  4854 ;
                                0EAD  4855 ; Functional Description:
                                0EAD  4856 ;
                                0EAD  4857 ;       This routine zero fills an incomplete MSCP message.
                                0EAD  4858 ;
                                0EAD  4859 ; Inputs:
                                0EAD  4860 ;
                                0EAD  4861 ;       R1      size of MSCP message
                                0EAD  4862 ;       R2      base address of MSCP message
                                0EAD  4863 ;
                                0EAD  4864 ; Outputs:
                                0EAD  4865 ;
                                0EAD  4866 ;       All registers preserved.
                                0EAD  4867 ;--
                                0EAD  4868
                                0EAD  4869 DUTU$FILL_MSCP_MSG::
                                0EAD  4870
                          3F    BB  0EAD  4871          PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Save registers.
                50  30    51    C3  0EAF  4872          SUBL3   R1, #MSCP$K_LEN, R0             ; Compute fill size.
                          07    15  0EB3  4873          BLEQ    90$                             ; Branch if nothing to fill.
          6241  50  00  6E    00    2C  0EB5  4874          MOVC5   #0, (SP), #0, R0, (R2)[R1]      ; Zero fill the message.
                          3F    BA  0EBC  4875 90$:     POPR    #^M<R0,R1,R2,R3,R4,R5>          ; Restore registers.
                          05    BE  0EBE  4876          RSB                                     ; Return
                                0EBF  4877
                                0EBF  4878          .END
```

```
$$BASE                          = 00000000
$$BIGDD                         = 00000010 R      06
$$BIGDT                         = 00000008 R      07
$$DISPL                         = 0000000C
$$GENSW                         = 00000001
$$HIGH                          = 0000000B
$$LIMIT                         = 0000000B
$$LOW                           = 00000000
$$MNSW                          = 00000001
$$MXSW                          = 00000001
ATE_ENTRY_LEN                   = 00000005
ATE_MSCPCODE                      00000002
ATE_OFFSET                        00000000
ATE_SSCODE                        00000003
BUG$_INCONSTATE                   ********    X   03
BUG$_IVDSKCONFG                   ********    X   03
BUG$_MSCPCLASS                    ********    X   03
CDDB$A_2PFKB                      00000174
CDDB$A_DAPCDRP                    00000194
CDDB$A_DAPIRP                     00000134
CDDB$A_PRMCDRP                    000000D0
CDDB$A_PRMIRP                     00000070
CDDB$B_CNTRLMDL                 = 00000026
CDDB$B_DAPCOUNT                 = 00000039
CDDB$B_SUBTYPE                  = 0000000B
CDDB$B_SYSTEMID                 = 0000000C
CDDB$B_TYPE                     = 0000000A
CDDB$K_DUTULENGTH               = 000001F8
CDDB$K_LENGTH                   = 00000070
CDDB$L_ALLOCLS                  = 00000050
CDDB$L_CANCLQBL                   000000B4
CDDB$L_CANCLQFL                   000000B0
CDDB$L_CDDBLINK                 = 00000058
CDDB$L_CDRPQBL                  = 00000004
CDDB$L_CDRPQFL                  = 00000000
CDDB$L_CDT                        000000F4
CDDB$L_CRB                      = 00000018
CDDB$L_DAPCDRP                  = 00000054
CDDB$L_DAPCDT                     00001B8
CDDB$L_DAPUCB                     00000150
CDDB$L_DDB                      = 0000001C
CDDB$L_ORIGUCB                  = 0000004C
CDDB$L_PDT                      = 00000014
CDDB$L_PRMUCB                     0000008C
CDDB$L_RSTRTQBL                 = 00000040
CDDB$L_RSTRTQFL                 = 0000003C
CDDB$L_SAVED_PC                 = 00000044
CDDB$L_UCBCHAIN                 = 00000048
CDDB$M_2PBSY                    = 00000800
CDDB$M_DAPBSY                   = 00000400
CDDB$M_INITING                  = 00000004
CDDB$M_NOCONN                   = 00000080
CDDB$M_POLLING                  = 00000020
CDDB$M_RECONNECT                = 00000008
CDDB$M_RESYNCH                  = 00000010
CDDB$M_SNGLSTRM                 = 00000001
CDDB$S_SYSTEMID                 = 00000006
CDDB$V_2PBSY                    = 0000000B
CDDB$V_DAPBSY                   = 0000000A
CDDB$V_RECONNECT                = 00000003
CDDB$W_CNTRLFLGS                = 00000028
CDDB$W_SIZE                     = 00000008
CDDB$W_STATUS                   = 00000012
CDDB_INIT_PRM_CDRP                00000159 R      03
CDRP$B_CARCON                   = FFFFFFDC
CDRP$B_CD_TYPE                  = 0000000A
CDRP$B_EFN                      = FFFFFFC2
CDRP$B_FIPL                     = 0000000B
CDRP$B_IRP_TYPE                 = FFFFFFAA
CDRP$B_PRI                      = FFFFFFC3
CDRP$B_RMOD                     = FFFFFFAB
CDRP$L_ABCNT                    = FFFFFFE0
CDRP$L_ABTDQBL                  = FFFFFFDC
CDRP$L_ABTDQFL                  = FFFFFFD8
CDRP$L_ARB                      = FFFFFFF8
CDRP$L_AST                      = FFFFFFB0
CDRP$L_ASTPRM                   = FFFFFFB4
CDRP$L_BCNT                     = FFFFFFD2
CDRP$L_CANIOCDRP                = FFFFFFA0
CDRP$L_CANIOQBL                 = FFFFFFA4
CDRP$L_CANIOQFL                 = FFFFFFA0
CDRP$L_CAN_RSPID                = FFFFFFF0
CDRP$L_CDT                      = 00000024
CDRP$L_DIAGBUF                  = FFFFFFEC
CDRP$L_DUTUFLAGS                = 00000040
CDRP$L_EXTEND                   = FFFFFFF4
CDRP$L_FQBL                     = 00000004
CDRP$L_FQFL                     = 00000000
CDRP$L_IOQBL                    = FFFFFFA4
CDRP$L_IOQFL                    = FFFFFFA0
CDRP$L_IOSB                     = FFFFFFC4
CDRP$L_IOST1                    = FFFFFFD8
CDRP$L_IOST2                    = FFFFFFDC
CDRP$L_JNL_SEQNO                = FFFFFFE8
CDRP$L_LBUFH_AD                 = 0000002C
CDRP$L_MEDIA                    = FFFFFFD8
CDRP$L_MSG_BUF                  = 0000001C
CDRP$L_OBCNT                    = FFFFFFE4
CDRP$L_PID                      = FFFFFFAC
CDRP$L_RSPID                    = 00000020
CDRP$L_RWCPTR                   = 00000028
CDRP$L_SEGVBN                   = FFFFFFE8
CDRP$L_SEQNUM                   = FFFFFFF0
CDRP$L_SNDABTQBL                = FFFFFFE4
CDRP$L_SNDABTQFL                = FFFFFFE0
CDRP$L_SVAPTE                   = FFFFFFCC
CDRP$L_TT_TERM                  = FFFFFFDC
CDRP$L_UCB                      = FFFFFFBC
CDRP$L_WIND                     = FFFFFFB8
CDRP$M_CAND                     = 00000001
CDRP$M_CANIO                    = 00000002
CDRP$M_IVCMD                    = 00000100
CDRP$M_PERM                     = 00000008
CDRP$Q_NT_PRVMSK                = FFFFFFE0
```

```
CDRP$V_CAND                     =  00000000
CDRP$V_CANIO                    =  00000001
CDRP$V_ERLIP                    =  00000002
CDRP$V_HIRT                     =  00000004
CDRP$V_IVCMD                    =  00000008
CDRP$V_PERM                     =  00000003
CDRP$W_ABCNT                    =  FFFFFFE0
CDRP$W_BCNT                     =  FFFFFFD2
CDRP$W_BOFF                     =  FFFFFFD0
CDRP$W_CDRPSIZE                 =  00000008
CDRP$W_CHAN                     =  FFFFFFC8
CDRP$W_DUTUCNTR                 =  00000044
CDRP$W_ENDMSGSIZ                =  00000046
CDRP$W_FUNC                     =  FFFFFFC0
CDRP$W_IRP_SIZE                 =  FFFFFFA8
CDRP$W_OBCNT                    =  FFFFFFE4
CDRP$W_STS                      =  FFFFFFCA
CDT$L_AUXSTRUC                  =  0000005C
COM$DRVDEALMEM                     ********    X   03
CRB$L_AUXSTRUC                  =  00000010
CRB$L_DUETIME                   =  00000018
CRB$L_INTD                      =  00000024
CRB$L_TOUTROUT                  =  0000001C
DAP_COUNT                       =  0000000A
DC$_TAPE                        =  00000002
DDB$K_LENGTH                    =  00000044
DDB$L_2P_UCB                    =  00000040
DDB$L_ALLOCLS                   =  0000003C
DDB$L_CONLINK                   =  00000038
DDB$L_DDT                       =  0000000C
DDB$L_LINK                      =  00000000
DDB$L_SB                        =  00000034
DDB$L_UCB                       =  00000004
DDB$S_NAME                      =  00000010
DDB$T_NAME                      =  00000014
DDT$L_MNTVER                    =  00000020
DEANONPAGED                        0000014F  R     03
DEV$M_2P                        =  00000010
DEV$M_AVL                       =  00040000
DEV$M_CDP                       =  00000008
DEV$V_2P                        =  00000004
DEV$V_AVL                       =  00000012
DEV$V_MSCP                      =  00000005
DIR...                          =  00000001
DO_ORIG_UCB                        00000232  R     03
DT$_RA60                        =  00000016
DT$_RCF25                       =  00000018
DT$_RD51                        =  00000019
DT$_RX02                        =  0000000B
DT$_TU81                        =  00000008
DUSCANCEL_FROM_HIRT                ********WL  GX  03
DUSRSTRTQ_HIRT_CDRP                ********WL  GX  03
DUSTEST_HIRT_RQAITCNT              ********WL  GX  03
DUPLICATE_SYSTEMID                 00000115  R     03
DUTU$AB_DEF_DEVTYPE                00000000  RG    07
DUTU$AW_DEF_DEVNAM                 00000000  RG    06
DUTU$BUILD_CANIO_CDRP              000009D4  R     03
DUTU$CANCEL                        000008BE  RG    03
DUTU$CANCEL_RDT                    00000A62  R     03
DUTU$CANCEL_RDTWAIT                00000A42  R     03
DUTU$CHECK_NOCANCEL                000009A1  R     03
DUTU$CHECK_RWAITCNT                00000D6F  RG    03
DUTU$CLEANUP_CANCEL                00000B02  R     03
DUTU$CREATE_CDDB                   00000000  RG    03
DUTU$DATA                          00000000  RG    02
DUTU$DEALLOC_ALL                   00000CC8  RG    03
DUTU$DEALLOC_RSPID_MSG             00000CD3  RG    03
DUTU$DEVTYPE_TABLE                 00000000  RG    08
DUTU$DISCONNECT_CANCEL             00000AE4  RG    03
DUTU$DODAP                         00000E0D  RG    03
DUTU$DRAIN_CDDB_CDRPQ              00000C6D  RG    03
DUTU$DUMP_COMMAND                  00000E83  RG    03
DUTU$DUMP_ENDMESSAGE               00000E92  RG    03
DUTU$END_CANCEL                    00000AFC  R     03
DUTU$FAILOVER_IODB                 000003AD  R     03
DUTU$FAILOVER_UCB                  0000030E  RG    03
DUTU$FILL_MSCP_MSG                 00000EAD  RG    03
DUTU$FIND_DDB                      00000632  RG    03
DUTU$GET_DEVNAM                    00000775  R     03
DUTU$GET_DEVTYPE                   000007F8  RG    03
DUTU$INIT_CONN_UCB                 00000825  RG    03
DUTU$INIT_MSCP_MSG                 00000B6F  RG    03
DUTU$INIT_MSCP_MSG_UNIT            00000B73  RG    03
DUTU$INSERT_RESTARTQ               00000C0F  RG    03
DUTU$INTR_ACTION_N                 00000BBC  RG    03
DUTU$INTR_ACTION_XFER              00000BB7  RG    03
DUTU$KILL_THIS_THREAD              00000D65  RG    03
DUTU$LINK_SEC_UCB                  000006DE  RG    03
DUTU$LINK_UCB2CDDB                 0000070F  RG    03
DUTU$LOG_IVCMD                     00000DAF  RG    03
DUTU$LOOKUP_UCB                    0000055B  RG    03
DUTU$L_CDDB_LISTHEAD               00000000
DUTU$NEW_UNIT                      00000441  RG    03
DUTU$POLL_FOR_UNITS                0000016F  RG    03
DUTU$POST_CDRP                     00000CE7  RG    03
DUTU$RECONN_LOOKUP                 00000C4C  RG    03
DUTU$RESET_MSCP_MSG                00000B55  RG    03
DUTU$RESTORE_CREDIT                00000BED  RG    03
DUTU$SEND_DRIVER_MSG               00000B92  RG    03
DUTU$SEND_DUPLICATE_UNIT           00000E6A  RG    03
DUTU$SEND_MSCP_MSG                 00000B9B  RG    03
DUTU$SETUP_CDP_UCB                 00000833  R     03
DUTU$SETUP_DUAL_PATH               0000057B  RG    03
DUTU$SEVER_CDDB                    00000756  RG    03
DUTU$SEVER_SEC_UCB                 00000737  RG    03
DUTU$TEMPLATE_ORB                  00000000  RG    05
DUTU$TEMPLATE_UCB                  00000000  RG    04
DUTU$TERMINATE_PENDING             00000C80  RG    03
DUTU$TEST_CANCEL_CDRP              00000B3B  RG    03
DUTU$TEST_CANCEL_DONE              00000AC5  RG    03
DUTU$UNITINIT                      000002F9  RG    03
DYN$C_CDRP                      =  00000039
DYN$C_CD_CDDB                   =  00000001
DYN$C_CLASSDRV                  =  00000064
```

```
DYN$C_FRK                       =  00000008
DYN$C_IRP                       =  0000000A
EMB$C_IVCMD                     =  00000007
END_ORIG_UCB_IO                    000002BF R       03
ERL$LOGMESSAGE                     ********   X     03
EXE$ALONONPAGED                    ********   X     03
EXE$DEANONPAGED                    ********   X     03
EXE$FORK_WAIT                      ********   X     03
EXE$GQ_SYSTIME                     ********   X     03
EXE$SNDEVMSG                       ********   X     03
FINISH_ORIGUCB                     000001E8 R       03
FKB$B_FIPL                      =  0000000B
FKB$B_TYPE                      =  0000000A
FKB$K_LENGTH                    =  00000018
FKB$W_SIZE                      =  00000008
INTR_ACTION_COMMON                 00000BBE R       03
IO$S_FCODE                      =  00000006
IO$V_FCODE                      =  00000000
IO$_PACKACK                     =  00000008
IO$_PHYSICAL                    =  0000001F
IO$_SETCHAR                     =  0000001A
IO$_SETMODE                     =  00000023
IOC$COPY_UCB                       ********   X     03
IOC$DELETE_UCB                     ********   X     03
IOC$GL_DEVLIST                     ********   X     03
IOC$GL_MUTEX                       ********   X     03
IOC$GL_PSBL                        ********   X     03
IOC$INITIATE                       ********   X     03
IOC$LINK_UCB                       ********   X     03
IOC$RETURN                         ********   X     03
IOC$SEVER_UCB                      ********   X     03
IOC$THREADCRB                      ********   X     03
IPL$_IOPOST                     =  00000004
IPL$_SCS                        =  00000008
IRP$B_CARCON                    =  0000003C
IRP$B_CD_TYPE                   =  0000006A
IRP$B_EFN                       =  00000022
IRP$B_FIPL                      =  0000006B
IRP$B_PRI                       =  00000023
IRP$B_RMOD                      =  0000000B
IRP$B_TYPE                      =  0000000A
IRP$C_LENGTH                    =  000000C4
IRP$K_LENGTH                    =  000000C4
IRP$L_ABCNT                     =  00000040
IRP$L_ABTDQBL                   =  0000003C
IRP$L_ABTDQFL                   =  00000038
IRP$L_ARB                       =  00000058
IRP$L_AST                       =  00000010
IRP$L_ASTPRM                    =  00000014
IRP$L_BCNT                      =  00000032
IRP$L_CANIOCDRP                 =  00000000
IRP$L_CANIOQBL                  =  00000004
IRP$L_CANIOQFL                  =  00000000
IRP$L_CAN_RSPID                 =  00000050
IRP$L_CDT                       =  00000084
IRP$L_DIAGBUF                   =  0000004C
IRP$L_DUTUFLAGS                 =  000000A0
IRP$L_EXTEND                    =  00000054
IRP$L_FQBL                      =  00000064
IRP$L_FQFL                      =  00000060
IRP$L_IOQBL                     =  00000004
IRP$L_IOQFL                     =  00000000
IRP$L_IOSB                      =  00000024
IRP$L_IOST1                     =  00000038
IRP$L_IOST2                     =  0000003C
IRP$L_JNL_SEQNO                 =  00000048
IRP$L_MEDIA                     =  00000038
IRP$L_OBCNT                     =  00000044
IRP$L_PID                       =  0000000C
IRP$L_RWCPTR                    =  00000088
IRP$L_SEGVBN                    =  00000048
IRP$L_SEQNUM                    =  00000050
IRP$L_SNDABTQBL                 =  00000044
IRP$L_SNDABTQFL                 =  00000040
IRP$L_SVAPTE                    =  0000002C
IRP$L_TT_TERM                   =  0000003C
IRP$L_UCB                       =  0000001C
IRP$L_WIND                      =  00000018
IRP$M_DIAGBUF                   =  00000080
IRP$M_FUNC                      =  00000002
IRP$M_MVIRP                     =  00002000
IRP$M_PHYSIO                    =  00000100
IRP$Q_NT_PRVMSK                 =  00000040
IRP$V_BUFIO                     =  00000000
IRP$W_ABCNT                     =  00000040
IRP$W_BCNT                      =  00000032
IRP$W_BOFF                      =  00000030
IRP$W_CDRPSIZE                  =  00000068
IRP$W_CHAN                      =  00000028
IRP$W_FUNC                      =  00000020
IRP$W_OBCNT                     =  00000044
IRP$W_SIZE                      =  00000008
IRP$W_STS                       =  0000002A
IVCMD_ALIGN                        00000054
IVCMD_ENDMSG                       00000000
IVCMD_ENDSIZ                    =  00000030
IVCMD_MSGLEN                       00000054
IVCMD_ORGMSG                       00000030
IVCMD_WORK                         00000054
LINK_2P_UCB                        000005D2 R       03
LINK_NEW_UCB                       000004A9 R       03
LOOKUP_LOCAL_UCB                   00000885 R       03
MSCP$B_OPCODE                   =  00000008
MSCP$K_CM_EMULA                 =  00000004
MSCP$K_CM_HSC50                 =  00000001
MSCP$K_CM_RC25                  =  00000003
MSCP$K_CM_RDRX                  =  00000007
MSCP$K_CM_TU81                  =  00000005
MSCP$K_CM_UDA50                 =  00000002
MSCP$K_CM_UDA52                 =  00000006
MSCP$K_EMB_OLD                  =  00000000
MSCP$K_LEN                      =  00000030
MSCP$K_MXCMDLEN                 =  00000024
MSCP$K_OP_ABORT                 =  00000001
```

```
MSCP$K_OP_DTACP     =  0000000B
MSCP$K_OP_GTUNT     =  00000003
MSCP$K_ST_AVLBL     =  00000004
MSCP$K_ST_DRIVE     =  0000000B
MSCP$K_ST_OFFLN     =  00000003
MSCP$K_ST_SUCC      =  00000000
MSCP$L_CMD_REF      =  00000000
MSCP$L_MEDIA_ID     =  0000001C
MSCP$L_OUT_REF      =  0000000C
MSCP$M_CF_ATTN      =  00000080
MSCP$M_CF_MISC      =  00000040
MSCP$M_CF_THIS      =  00000010
MSCP$M_EU_CTYPE     =  00000F00
MSCP$M_EU_DESIG     =  00007000
MSCP$M_EU_SUBC      =  000000F8
MSCP$M_MD_NXUNT     =  00000001
MSCP$M_SHADOW       =  00008000
MSCP$M_ST_MASK      =  0000001F
MSCP$S_EU_DESIG     =  00000003
MSCP$S_MTYP_DO      =  00000005
MSCP$S_MTYP_D1      =  00000005
MSCP$S_ST_MASK      =  00000005
MSCP$V_EU_DESIG     =  0000000C
MSCP$V_MTYP_DO      =  0000001B
MSCP$V_MTYP_D1      =  00000016
MSCP$V_SC_NOVOL     =  00000005
MSCP$V_SHADOW       =  0000000F
MSCP$V_ST_MASK      =  00000000
MSCP$W_MODIFIER     =  0000000A
MSCP$W_STATUS       =  0000000A
MSCP$W_UNIT         =  00000004
MSCP$W_UNT_FLGS     =  0000000E
MSCP_BOF            =  0000005C
MSG$_DUPUNITNO      =  00000058
MTX$W_OWNCNT        =  00000000
OH_NO               =  00000155 R       03
PCB$L_PID           =  00000060
PDT$L_ALLOCMSG      =  00000014
PDT$L_DEALLOMSG     =  00000020
PDT$L_MAXBCNT       =  000000BC
PDT$L_RCHMSGBUF     =  00000044
PDT$L_RCLMSGBUF     =  00000048
PDT$L_SNDCNTMSG     =  00000060
PDT$L_UNMAP         =  00000064
POLL_CONN_BROKE     =  00000220 R       03
POLL_LOOP           =  00000186 R       03
PR$_IPL             =  00000012
PR$_SIRR            =  00000014
RECONN_EXIT         =  00000C7F R       03
RESTART_POLL        =  00000223 R       03
SAVABS...           =  00000074
SB$L_DDB            =  00000054
SCS$ALLOC_RSPID     =  ********    X    03
SCS$DEALL_RSPID     =  ********    X    03
SCS$LKP_RDTCDRP     =  ********    X    03
SCS$LKP_RDTWAIT     =  ********    X    03
SCS$RECVL_RSPID     =  ********    X    03
SCS$UNSTALLUCB      =  ********    X    03
SEND_ABORTS         =  00000963 R       03
SS$_CANCEL          =  00000830
SS$_MEDOFL          =  000001A4
SS$_NORMAL          =  00000001
SS$_OPINCOMPL       =  000002D4
SS$_VOLINV          =  00000254
SVCDRP              =  00000068
SVERLIP             =  00000058
SVPDT               =  00000064
SVR7                =  0000006C
SVRET               =  00000070
SVSTATUS            =  00000054
SVUCB               =  00000060
SYS$GL_BOOTUCB      =  ********    X    03
SYS$GL_OPRMBX       =  ********    X    03
UCB$B_DEVCLASS      =  00000040
UCB$B_DEVTYPE       =  00000041
UCB$B_FIPL          =  0000000B
UCB$L_2P_ALTUCB     =  000000A8
UCB$L_2P_CDDB       =  000000C0
UCB$L_2P_DDB        =  000000A0
UCB$L_2P_LINK       =  000000A4
UCB$L_CDDB          =  000000BC
UCB$L_CDDB_LINK     =  000000C4
UCB$L_CDT           =  000000C8
UCB$L_CRB           =  00000024
UCB$L_DDB           =  00000028
UCB$L_DDT           =  00000088
UCB$L_DEVCHAR       =  00000038
UCB$L_DEVCHAR2      =  0000003C
UCB$L_DPC           =  0000009C
UCB$L_IOQFL         =  0000004C
UCB$L_LINK          =  00000030
UCB$L_MAXBCNT       =  000000B4
UCB$L_MAXBLOCK      =  000000B0
UCB$L_MEDIA_ID      =  0000008C
UCB$L_ORB           =  0000001C
UCB$L_PDT           =  00000084
UCB$L_RECORD        =  000000B0
UCB$L_STS           =  00000064
UCB$M_DELETEUCB     =  00010000
UCB$M_MSCP_FLOVR    =  00000800
UCB$M_MSCP_INITING  =  00000200
UCB$M_MSCP_WAITBMP  =  00000400
UCB$M_ONLINE        =  00000010
UCB$Q_UNIT_ID       =  000000CC
UCB$S_UNIT_ID       =  00000008
UCB$V_DELETEUCB     =  00000010
UCB$V_MSCP_FLOVR    =  0000000B
UCB$V_MSCP_INITING  =  00000009
UCB$V_MSCP_MNTVERIP =  00000008
UCB$V_MSCP_PKACK    =  0000000C
UCB$V_MSCP_WAITBMP  =  0000000A
UCB$V_ONLINE        =  00000004
UCB$V_POWER         =  00000005
UCB$V_TU_SEQNOP     =  00000002
```

```
UCB$V_VALID                     = 0000000B
UCB$W_DEVSTS                    = 00000068
UCB$W_MSCPUNIT                  = 000000D4
UCB$W_REFC                      = 0000005C
UCB$W_RWAITCNT                  = 00000056
UCB$W_STS                       = 00000064
UCB$W_UNIT                      = 00000054
UCB$W_UNIT_FLAGS                = 000000E0
VEC$L_IDB                       = 00000008
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000001F8 (504.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$220_DUTU_DATA_00	00000000 (0.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$115_DRIVER	00000EBF (3775.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$200_TEMPLATE_UCB_00	00000000 (0.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$200_TEMPLATE_ORB_00	00000000 (0.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$220_DEF_DEVNAM_TABLE	00000018 (24.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$220_DEF_DEVTYPE_TABLE	0000000C (12.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$220_DEVTYPE_TABLE_00	00000000 (0.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$220_DEVTYPE_TABLE_02	00000019 (25.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.04	00:00:01.23
Command processing	108	00:00:00.45	00:00:04.37
Pass 1	867	00:00:26.62	00:01:36.38
Symbol table sort	0	00:00:03.53	00:00:13.71
Pass 2	403	00:00:08.38	00:00:29.70
Symbol table output	1	00:00:00.26	00:00:01.40
Psect synopsis output	0	00:00:00.04	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1410	00:00:39.32	00:02:26.83

```
The working set limit was 2850 pages.
232553 bytes (455 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3308 non-local and 162 local symbols.
4878 source lines were read in Pass 1, producing 39 object records in Pass 2.
73 pages of virtual memory were used to define 68 macros.
```

```
                                    +----------------------------+
                                    ! Macro library statistics !
                                    +----------------------------+

Macro library name                                  Macros defined
------------------                                  --------------
_$255$DUA28:[DRIVER.OBJ]DUTULIB.MLB;1                       10
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                              37
_$255$DUA28:[SYSLIB]STARLET.MLB;2                           12
TOTALS (all libraries)                                      59

3624 GETS were required to define 59 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:DUTUSUBS/OBJ=OBJ$:DUTUSUBS MSRC$:DUTUSUBS/UPDATE=(ENH$:DUTUSUBS)+EXECML$/LIB+LIB$:DUTULIB/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DYDRIVER
LIS

OUTUEND
LIS

DUTUSUBS
LIS

DXPDRIVER
LIS

DXUTILITY
LIS

LADRIVER
LIS